DESIGN AUTOMATION TOOLS FOR A FPAA SOC AND ITS APPLICATION IN ANALOG EDUCATION

A Dissertation
Presented to
The Academic Faculty

by

Michelle D. Collins

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2016

DESIGN AUTOMATION TOOLS FOR A FPAA SOC AND ITS APPLICATION IN ANALOG EDUCATION

Approved by:

Professor Jennifer Hasler, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Linda Wills
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Bonnie Ferri
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Mark Smith
School of Information and Communication Technology
KTH Royal Institute of Technology

Professor Sung Kyu Lim
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 21 October 2016

To my loving family and supportive friends.

I appreciate you all for your time, advice, and encouragement!

I am blessed to be surrounded by a village that nurtures me...

ACKNOWLEDGEMENTS

I first want to give praise to God for being the source of my strength and seeing me through the completion of this degree. I am grateful for the guidance and assistance received during my time at Georgia Tech. I would like to thank these individuals for their kindness and sincere rapport. My advisor Dr. Jennifer Halser shared with me an interest in students and education that led to this dissertation. She offered her thoughts and I learned a lot while working alongside her both academically and personally. I am thankful for the opportunities she afforded me. I want to acknowledge my gracious committee Dr. Bonnie Ferri, Dr. Sung Kyu Lim, Dr. Linda Wills, and Dr. Mark Smith for providing me invaluable feedback. I appreciate the camaraderie of my ICEmates and will cherish our moments together. I want to thank my accountability group and Sloan peeps for the good vibes and for establishing a safe space with me to learn from each other. Last and surely not least, I want to thank my family for their love, encouragement, time, and faith. Their presence in my life enabled me to focus on my education intently.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY

ADC Analog-to-Digital Converter. An electronic system that converts an analog signal to a digital signal.

ASIC Application Specific Integrated Circuit. An integrated circuit custom-built for a special application and not for general purpose use.

AWG Arbitrary Waveform Generator. A DAC sampling a provided input voltage vector at a rate chosen by user.

BLIF Berkeley Logic Interchange Format. Is used to describe the hierarchy of a circuit in textual form or netlist.

BPF Band Pass Filter. An electronic device or circuit that allows signals with frequencies within a certain range and attenuates signals with frequencies outside that range.

C block Connection block. A crossbar global routing architecture module in FPAA hardware.

C4 Capacitively-Coupled Current Conveyer. A continuous time band pass filter.

CAB Computational Analog Block. A fundamental building block of analog technology like FPAAs that has basic analog components, which can be configured by end-users.

CAD Computer-Aided Design. The use of computer technology to enable the development, optimization, documentation, or analysis of a design.

CLB Computational Logic Block. A fundamental building block of digital technology like FGPAs that has basic logic components, which can be configured by end-users.

CMOS Complementary Metal–Oxide Semiconductor. A semiconductor technology used for making integrated circuits.

CPLD Complex Programmable Logic Device. It is used to build digital circuits.

DAC Digital-to-Analog Converter. An electronic system that converts a digital signal to an analog signal.

FG Floating Gate. For FPAAs, a pFET that has a capacitively coupled gate (electrically isolated) that has no path to DC ground, which allows it to hold charge.

FG OTA Floating Gate Operational Transconductance Amplifier. An amplifier with FG differential input voltages that produces an output current; voltage controlled current source (VCCS).

FIR Finite Impulse Response. A filter with a finite impulse response.

FPAA Field Programmable Analog Array. Programmable and configurable analog hardware.

FPAADD Field Programmable Array of Analog and Digital Devices. A hybrid combination of a FPAA and a FPGA.

FPGA Field Programmable Gate Array. Programmable and configurable digital hardware.

GUI Graphical User Interface. User interface that is based on graphical icons interaction and not typed command labels or text navigation.

IC Integrated Circuit. A set of electronic circuits fabricated in silicon.

I/O Input/Output pad. An intermediate structure connecting internal signals from an integrated circuit to the outside world.

LPF Low Pass Filter. An electronic device or circuit that allows signals of a low frequency range.

MIPS Microprocessor without Interlocked Pipeline Stages. A reduced instruction set computer (RISC) instruction set architecture (ISA).

MMAC Million Multiply-Accumulate. A unit for the number of multiply-accumulate operations performed.

MSP430 A Texas Instruments (TI) open source microprocessor.

OTA Operational Transconductance Amplifier. An amplifier with differential input voltages that produces an output current; a voltage controlled current source (VCCS).

PCB Printed Circuit Board. It connects electronic components using conductive traces, through-silicon vias (TSVs), pads, and other structures on a non-conductive substrate.

PLD Programmable Logic Device. It is used to build small digital circuits.

RASP Reconfigurable Analog Signal Processor. A type of FPAA that uses floating gate (FG) pFETs for routing switches, local memory, and programmable current sources.

RASP Tools A software suite for designing and simulating analog, digital, and mixed-signal circuits and systems as well as programming these designs on configurable hardware (FPAA SoCs) for experimental testing.

S block Switch block. A diagonally connected global routing architecture module in FPAA hardware.

sci2blif Scilab to BLIF. A tool that translates a user Xcos design to its corresponding BLIF file.

Scilab An open source computational software package and a high-level programming language.

scs_m Scilab data structure that contains all the information of a Xcos diagram that contains a user's design.

SNR Signal-to-Noise Ratio. A measure of signal strength relative to environment noise.

SoC System on Chip. An IC that contains within a single chip all the components of an electronic system.

SPICE An analog electronic circuit simulator environment.

Verilog A hardware description language used to design, model, and document electronic systems.

VLSI Very-Large-Scale Integration. A process to produce an integrated circuit with thousands of transistors within a single chip.

VMM Vector Matrix Multiplier. An analog solution for computing vector-matrix multiplications via current summation.

VPR Versatile Place and Route. An open source CAD tool that was developed to use a technology (architecture) mapped netlist to pack, place, and route a circuit onto a FPGA.

vpr2swcs VPR to switches. A tool that translates a FPAA component mapped BLIF file into a list of FG switches. Where these switches need to programed on or to a particular value to create the circuit designed by the user.

VTR Verilog-to-Routing. An open source CAD tool that maps a circuit described in Verilog to a specified FPGA architecture.

WTA Winner-Take-All. A circuit that essentially computes the max function of its inputs such that the output of the highest input is a low voltage and the other outputs are at a high voltage.

x2c Xcos to Chip. A tool that translates a user Xcos design to a FPAA SoC IC switch list file to be programmed.

Xcos An open source graphical editor within Scilab for constructing and simulating circuits and systems.

SUMMARY

The objective of this research is to emphasize the use of configurable and programmable mixed-signal architectures and a concomitant software suite in analog education. In particular this work focuses on a Field Programmable Analog Array (FPAA) System on Chip (SoC) and RASP Tools, respectively. The prevalence of digital design among diverse individuals in academia, industry, and the public community is recognized worldwide. Some of the preeminence that digital systems have experienced is attributed to Field Programmable Gate Arrays (FPGAs) and their tools. It is postulated that the lack of equal mobility for analog design has discouraged individuals from choosing to incorporate analog circuits and systems in their projects, despite their advantages. The notion is that if designers have access to a similar set of hardware and an accompanying software infrastructure, they can be enabled to reap the benefits of using analog and mixed-signal circuits for a range of applications. The inner workings of RASP Tools, which is the result of tool integration from a graphical front-end design environment to the FPAA hardware is described in detail. Further, RASP Tools and FPAAs were employed in a graduate course and an assessment of its effectiveness was conducted, mainly the students' satisfaction of our approach and their proficiencies. The assessment results led to the conclusion that we are moving in the right direction to create a user-friendly way for students to become familiar with analog design. It should be noted that there is room for improvement in tool development as we strive to create a foundation that will foster a balanced design community through education.

CHAPTER I

IMPACT OF FPAA APPLICATIONS

Field Programmable Gate Arrays (FPGAs) are used by a diverse audience that vary in age, experience, and background. The user has the freedom to think of some logic, compile it on their computer using vendor supplied software, simulate the functionality using the same software, and download the resulting binary file after connecting the FPGA to their computer. There is no restriction on the number of times any design can be implemented on the FPGA using its fine grain resources, but it is necessary to reprogram after power is lost and restored. We can find consumers trying out projects like blinking a collection of LEDs that may or may not be linked with a timer or music as well as making doorbells and interfacing with other hardware to create a digital oscilloscope. The complexity of the designs, in general, is limited by the imagination of the individual, which invites both novices and experts to tinker and build their ideas. FPGA aficionados also benefit from the widespread acceptance (*e.g.*, academia and industry) and its supportive community because they are free to learn from each other and collaborate. Thus, the digital world has a means to attract and encourage new users as well as retain them.

After looking at the success of FGPAs and their impact on digital design, it would be easy to assume that the analog world shares equivalent mobility worldwide. This expectation on investigation proves to be wrong and in fact exposes that analog design is considered a niche area. Figure 1 depicts a disparity in the population of analog and digital designers. Unfortunately, there is an associated difficulty with classic analog design as there are fewer people that create analog designs when compared to the vast amount of digital designers at all expertise levels. It is reasonable to suggest

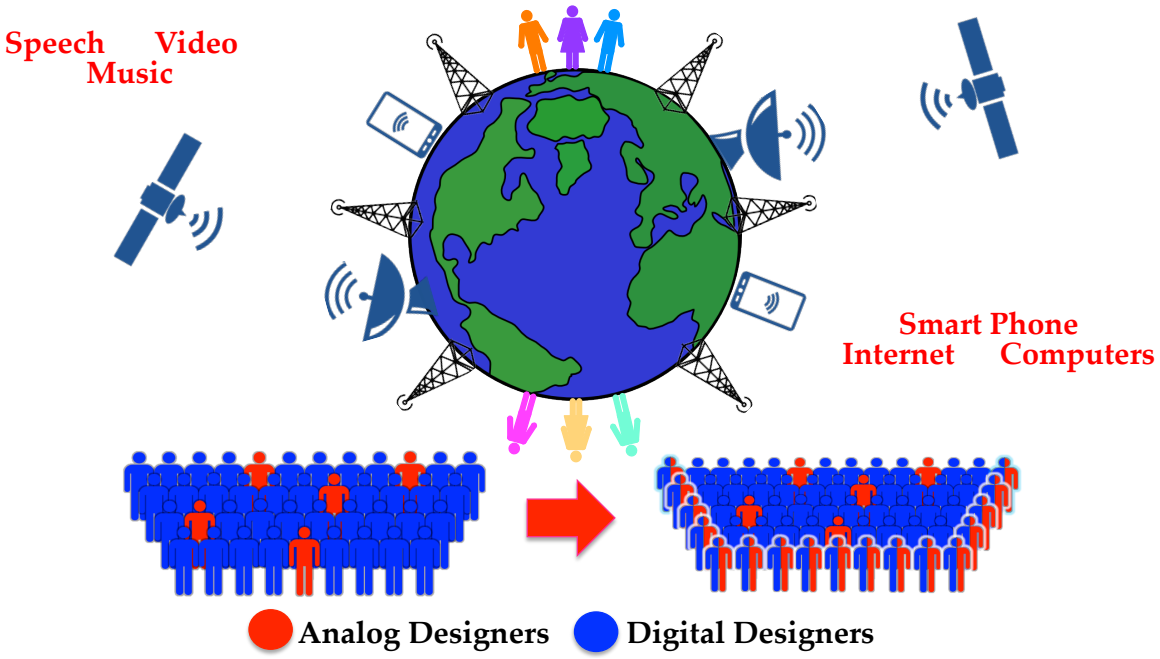


Figure 1: We live in an analog world, where we are surrounded by analog signals like sound, radio, light, *etc.* However, the way we interact with the world is mainly through digital interfaces like smart phones, computers, *etc.* This has led to a disproportionate ratio of analog to digital engineers as illustrated. However, as we strive to make devices smarter, smaller, and low-power to extend battery life, we need to consider analog solutions more often. The goal is to create a design community that is balanced with equal exposure to analog and digital systems. Where education is key to bring about this change.

that beginners would prefer digital over analog components in projects, which restricts the analog pipeline of newcomers and keeping them interested.

There is a need for analogous analog hardware and software to garner the level of progress seen with FPGAs. Some metrics to consider are learning curve, user friendliness, hardware and software capabilities, and community. Configurable analog hardware, namely Field Programmable Analog Arrays (FPAAs) and its complementary tool suite, fill this void. Unlike digital circuits that are binary, analog circuits can use the full power rail range for computation and consume considerably less power per operation [32]. Analog computation is well suited for low accuracy and number of bits while digital is better for high accuracy and number of bits. Thus, an FPAA that can combine both analog and digital components as programmable elements is

the most promising approach.

The objective of this research is to emphasize the use of configurable and programmable mixed-signal architectures and a concomitant software suite in analog education. In particular this work focuses on FPAA system on chips (SoCs) and RASP Tools, respectively. The notion is that if designers have access to a suitable set of hardware and an accompanying software infrastructure, they can be enabled to reap the benefits of using analog and mixed-signal circuits for a range of applications.

The approach that remains after finding a suitable FPAA, is creating an integral and flexible software suite that is accessible, uncomplicated, and capable of simulation. The next step to attain congruency with its digital counterpart will be to use this software and hardware in an environment with a target group of individuals to note and incorporate highly desirable and essential characteristics. This methodology aims to create the foundation that will lead to building a strong community in analog design as well as mixed-signal design.

1.1 FPAA and Tools

FPAAs are the remarkable devices positioned to spark an analog resurgence within the IC design community. A complementary software suite would support FPAAs in achieving this feat by enabling design automation at the device, circuit, and system level for users with varying experience. FPAAs and its tool suite have been iteratively designed to give users full access to circuit component staples and data path flow through available programmable features. Fabricated in well known complementary metal-oxide semiconductor (CMOS) processes, FPAAs are continuing to take full advantage of node technologies scaling to nanometer dimensions. There are promising results that the dwindling CMOS process feature size will not force FPAA hardware to relinquish its purpose to fill an aperture within the field of circuits and systems.

1.1.1 Evolution of FPAA SoC

Modern FPAAs are similar to FPGAs as they are both fabricated integrated circuits that can be configured by users. Both structures have multiple block modules with sub-elements that can be connected by a routing infrastructure. Previous FPAAs were once the analog equivalent of digital complex programmable logic devices (CPLDs) due their size and architecture. Figure 2 gives a visual comparison of these devices. CPLDs achieve logic functions using sum-of-product macrocells, while FPGAs contain digital components such as flip-flops and lookup tables. FPAAs on the other hand usually contain amplifiers and capacitors. It is practical to use CPLDs for simple combinational logic, but FPGAs for large state machines. However, there is now a class of FPAAs that have both analog and digital block modules for mixed-signal applications.

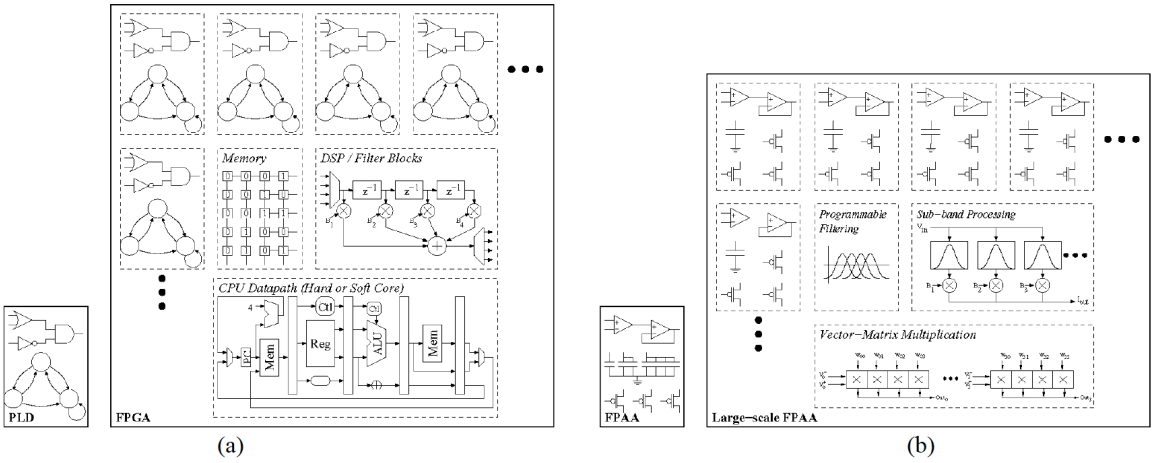


Figure 2: (a) Digital PLDs and CPLDs can be used to implement small modules of a complex system, while FPGAs can be used to implement entire systems. (b) Analogously, traditional FPAAs resemble the early PLDs and CPLDs as they are for small systems, but FPAAs based on floating gate devices can implement high-level systems. Image reproduced from [35].

FPAAs are more than standalone chips as they are equipped for integration into embedded systems. Power consumption is a major concern for embedded systems. When these systems are created the developer has to consider the trade off between employing the fastest circuits and maintaining the required cooling mechanisms that

will satisfy product specifications. One of the underlying reasons that FPAAs are very promising for low-power applications is that they are more than a decade ahead of DSP microprocessors in terms of power dissipation according to TI Fellow Gene Frantz's Law that complements Moore's Law [34]. Gene's Law postulates that the power consumption in digital signal processing (DSP) microprocessors, as measured in milliwatts per microprocessor without interlocked pipeline stages (mW/MIPS) as well as in milliwatts per million multiply-accumulates (mW/MMAC), is halved about every 18 months [29, 30]. The unit mW/MMAC is more appropriate when discussing power for an operation because mW/MIPS may contain misleading non-computation MIPS instructions like null operation (NOP). FPAAs have demonstrated a 10,000 improvement in this category over its digital counterpart [34].

FPAAs and FPGAs are well-suited for rapid prototyping and reducing time to market, while highly specialized application specific integrated circuit (ASIC) hardware is designed for a single application and has a long design cycle. FPAAs have a competitive edge over FPGAs in that they consume less power because they can implement low-power systems and easily realize digital solutions within the same platform. Over the last two decades multiple FPAAs have been made with the goal of bringing analog design to the forefront to be incorporated in more signal processing and sensing applications. Thus, the latest FPAA IC architecture, a complete system on chip (SoC), is the most mature and refined device produced yet [32].

Figure 3 shows a visual progression of FPAAs built at Georgia Tech (GT) and a few other FPAAs to provide time context. The instrumental programmable floating gate (FG) device plays several roles within these GT FPAAs: routing switches, local analog memory to store coefficients, accurate bias current sources, and behavior modifiers of in–circuit active members. The interconnect scheme as well as the programming infrastructure have been improved over the years. The circuit components in the computational analog blocks (CABs), the grouping of the elements, and the kinds of

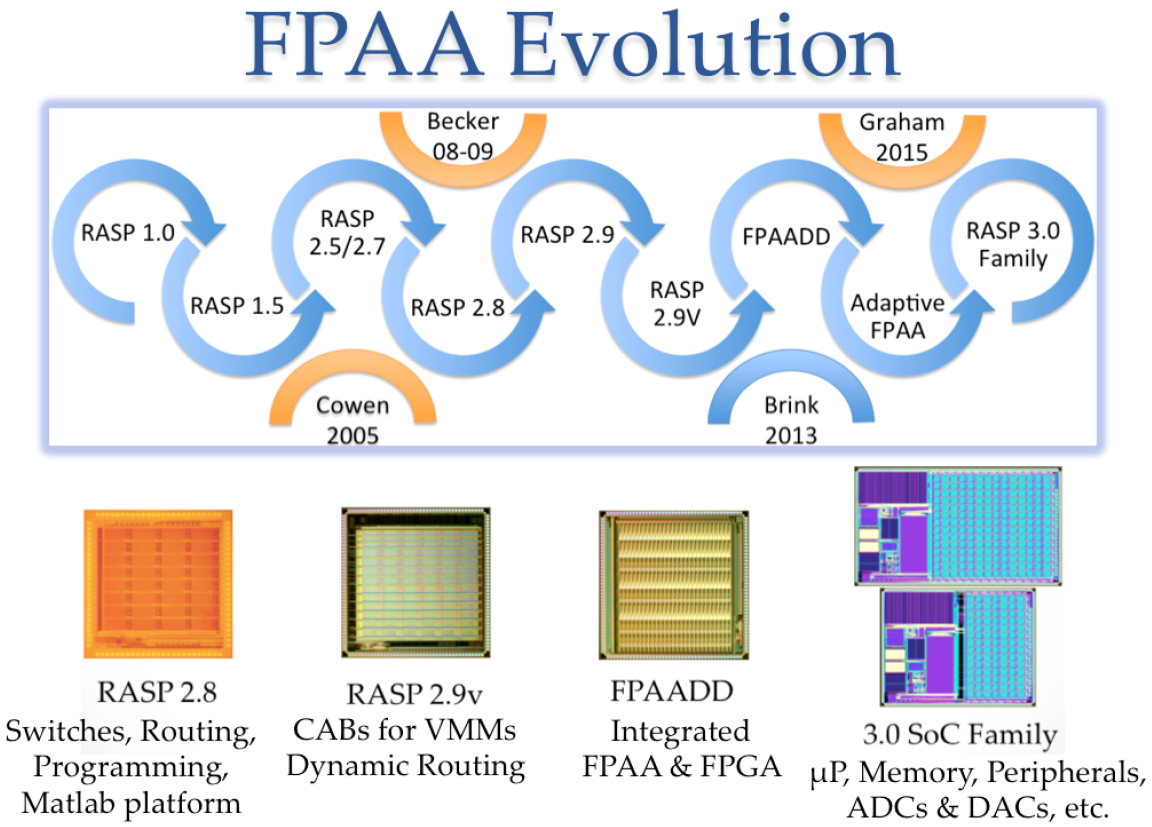


Figure 3: Trajectory of FPAAs created at Georgia Tech and select FPAAs from literature.

CABs vary across each architecture iteration. Similarly, computational logic blocks (CLBs) featured in later chips were modified to their present form.

1.1.1.1 Initial FPAAs

Hall [34] proposed the first functional Reconfigurable Analog Signal Processor (RASP 1.0) using FG devices as the solution for the short-comings of previous FPAAs in literature [16,49,50,60,62,75]. FG devices, pFETs with a capacitively coupled gate (*i.e.*, no DC path to a fixed potential), have innate attributes that increase the amount of analog computations calculated in an IC. This concept of large scale analog reconfigurable hardware required a repeatable module that would be placed in an array structure, similar to stamped die on a wafer at a macro level. This analogy does not convey the inherent interconnects between the reconfigurable modules that enable the

transfer of information between sub-elements. A CAB was well-suited for this task because it consisted of computational logic that easily tiled. The switch network was constituted of FG transistors, which alleviated the need for an excess digital memory quantity and the additional associated latency. The incorporated CABs had medium and coarse grain components that provided adaptability and optimization for signal processing such as: operational transconductance amplifiers (OTAs), FETs, fixed–value capacitors, capacitively coupled current conveyors (C^4), a 4 x 4 vector matrix multiplier (VMM), and a peak detector. Band pass filtering, sub-banding, Fourier processing, frequency decomposition, differentiation, and matrix transformations were all possible with this infrastructure [34].

The next major advancement of the RASP IC family was accomplished with the invention of the RASP 2.8 [6]. An improvement of >9-bits to represent FG dynamic range, an increased speed of accurately programming ≈20 gates/s, and a better distinction between ON and OFF switches were achieved. The programming algorithm was updated by moving the circuitry on-chip and communicating via a SPI digital interface. This architecture was fabricated in a 350nm CMOS process and had the dimensions 3×3 mm^2. The RASP 2.8 chip was organized into an array which contains two types of CABs that encompass a combination of programmable transconductors, multipliers to transistors, and capacitors. A total of 32 CABs were incorporated into the design in an alternating and uniform row fashion arranged in a 4×8 matrix array. This chip had 50,000 FGs that also doubled as programmable analog parameters to be used as switches, biased current sources, or local memory. Researchers created an AM receiver with a gilbert multiplier, comparators, a low pass filter, and a delta-sigma modulator without the use of a clock to successfully demodulate a 500-kHz carrier modulated by a 3-kHz triangle wave input. An analog speech processor that enhances the signal-to-noise ratio (SNR) for one sub-band of a noisy speech signal was also realized in this FPAA.

A few years later GT researchers addressed the demand for on-the-fly topology changes and the urgency for software to handle analog design using the large FPAA system. The RASP 2.9v manufactured in 350nm CMOS with an area of 25-mm^2 was the solution. The innovated IC had 78 CABs and the breakdown was 36 general purpose, 18 DACs, and 24 VMM CABs. It is different from previous FPAAs as it has a hybrid switch matrix and volatile switches. Within the Mathworks MATLAB platform, the chip was programmed and tested. An image transform system was implemented, which used a 3×3 Sobel edge detector on the first convolution pass and a 9×9 smoothing filter on the second pass. The RASP 2.9v was equipped to build an arbitrary waveform generator (AWG) using the volatile switches, an OTA, and a FG OTA; an added bonus of this IC was the ability to sum multiple programmed current mode waveforms to generate a new waveform. Lastly, a current mode mixed-signal FIR filter with a linear phase that had a total power of 12μW operating at 1-MHz was shown [68].

Prior to the creation of the field programmable array of analog and digital devices (FPAADD), FPAAs only contained analog elements for design. FPAADDs have an architecture that complements a digital system within one IC [85]. FPAADD's interchangeable digital (D) and analog (A) tiles were built from a CLB or a CAB, respectively. In this IC, the typical conversion from analog to digital and digital to analog via dedicated ADCs and DACs as well as a clear delineation between the two modes of processing was circumvented. Instead, these two modes of computation (*i.e.*, CAB and CLB) were interleaved within an array that also shares a common global heterogeneous Manhattan style interconnect scheme. An improvement over SRAM based FPAAs for memory was achieved as parasitics reduced. The FPAADD occupying a 5×5 mm^2 die area with over 130,000 FGs was made with 350nm CMOS technology. Applications like "built-in self test, digitally assisted analog computation, industrial control, machine learning, mixed-signal processing, digitally tunable

analog circuits, and biologically inspired neuromorphic circuits" were all feasible in the FPAADD [85]. A voltage controlled oscillator (VCO) based ADC and a 2^{nd} order $\Sigma - \Delta$ modulator were also explored.

1.1.1.2 FPAA SoC Family

The RASP 3.0 family of FPAA SoCs are the most sophisticated FPAAs built to date. The RASP 3.0 is a 12×7 mm^2 die fabricated in a 350nm CMOS process that has over 500,000 FGs [32]. The RASP 3.0 SoC combines approaches of earlier FPAAs and an open source MSP430 μP [59]. It also employs on-chip structures like 7-bit signal DACs, a ramp ADC, and memory-mapped I/Os. The culmination of creating new FPAAs with more capabilities has led to this FPAA SoC having an infrastructure that is situated within a co-design space between three domains (analog, digital, and μP). Figure 4 depicts the block diagram of the FPAA SoC with these components. Another feature of this SoC is configurability on the fly using a set of T-gate based switch elements in the routing fabric that is akin to the idea in [68]. VMMs are now readily formed within the routing resources when desired. A compiled ramp ADC and analog auditory word classifier have been demonstrated using this FPAA SoC.

1.1.2 FPAA SoCs and Co-design

The emergence of large-scale mixed-mode configurable systems, such as the FPAA SoC [32], exposes the need for tools that enable designers to effectively and efficiently solve the vast open questions of the analog–digital co-design space. Digital-only hardware-software co-design is a traditional, although unsolved and actively researched, discipline (*e.g.*, [84]); incorporating analog computation and signal processing adds a new dimension to co-design. Current research in hardware-software co-design focuses entirely on digital hardware-software (*e.g.*, processor) co-design (for example, [65, 82, 87]). Well-established FPGA design tools, such as Simulink [57], are developed to work with Xilinx [86] and Altera FPGA devices [2, 3]. Simulink, and

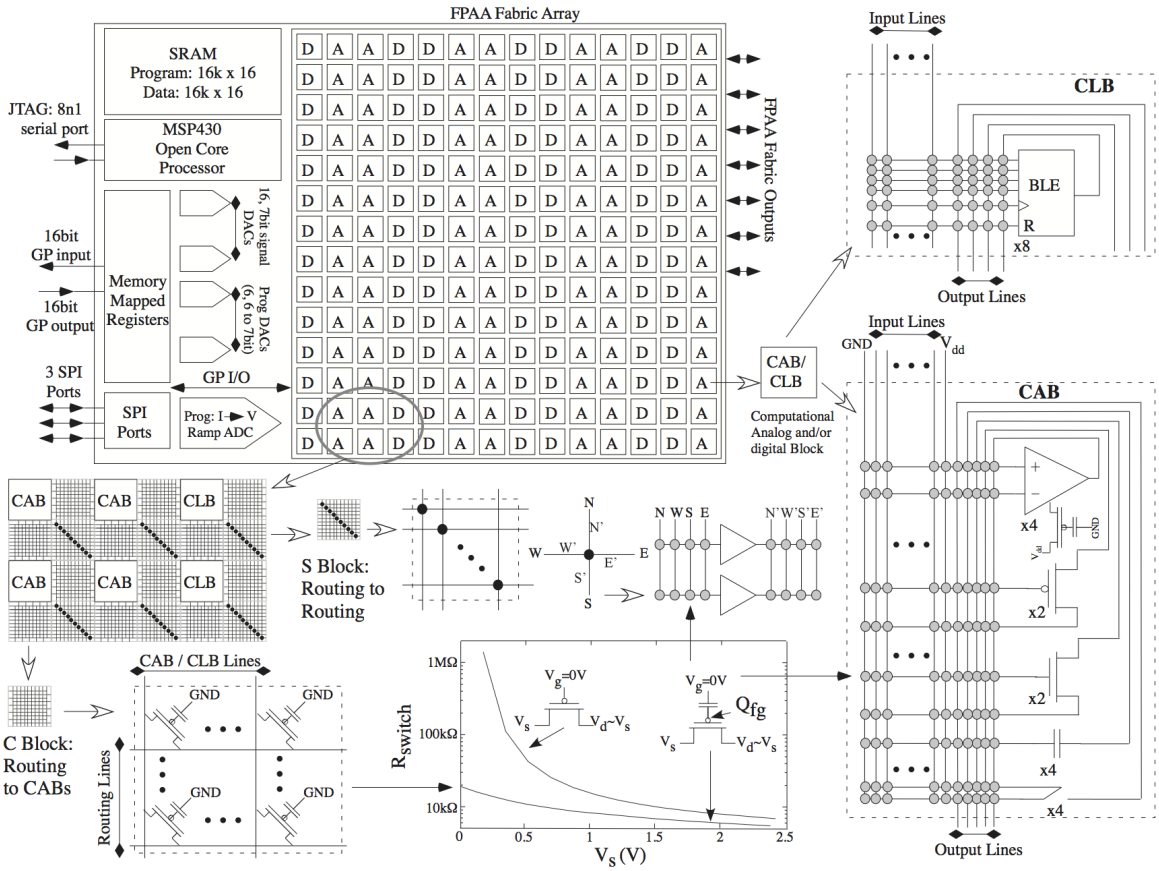


Figure 4: The block diagram illustrates the computational blocks and routing architecture. The RASP 3.0 integrates concepts from previous GT FPAAs [6,68,85]. This IC features an open source MSP430 processor [59], on-chip DACs and ADCs, general purpose I/Os, SRAM, current-to-voltage conversion, voltage measurement, and essential peripherals [32]. The FG switches in the connection (C) blocks, the switch (S) blocks, and the local routing consist of single pFET FG transistors programmed to be a closed switch over the entire fabric signal swing of 0–2.5 V [11]. Eight, four-input BLE lookup tables with a latch comprise the CLB tiles. Transconductance amplifiers, transistors, capacitors, switches, and other elements constitute the CAB tiles.

to a lesser extent some open source tools (*e.g.*, [71]), provides the framework for custom Xilinx and Altera compilation tools. Minute details are completely abstracted away from the user. These tools support both standard Simulink blocks to compile to Verilog blocks which are then mapped to targetable hardware as well as assist the creation of unique blocks devised for a specific hardware platform.

The wide demonstration of programmable and configurable analog signal processing and computation [32,69] revealed an additional range of accessible design choices.

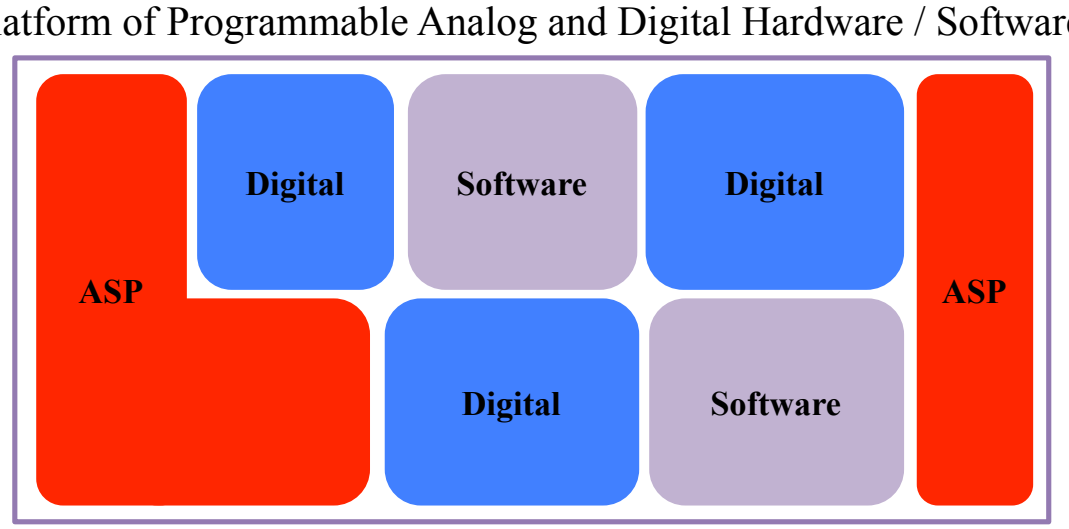


Figure 5: A mixed platform model of programmable resources that can transform an application to a heterogeneous set of analog and digital hardware + software elements. The programming of a particular FPAA system is defined by its unique technology files, chosen by the user, that encompass a combination of components with analog (*i.e.*, computational analog block (CAB)) or digital members (*i.e.*, computational logic block (CLB)) that are connected through a switch matrix as well as a range of I/Os and additional special devices.

The opportunity to explore analog-digital hardware-software co-design using FPAA SoCs requires user-friendly design tools to enable system design without requiring an understanding of analog circuit components. Occasionally, analog automation tools are discussed [23,31]; usually, these treatments are theoretical in nature because of the lack of available experimental hardware. Thus, the momentum required to develop working systems, including research lab or classroom demonstrations, are not empowered. However, the tool suite for FPAAs discussed in Chapter 2 of this thesis provides a starting point for the analog–digital software co-design discussion to further progress through an open source platform as larger future mixed-mode configurable systems are developed. The tool infrastructure facilitates users in manipulating design choices (*i.e.*, power, area) involving mixed-signal computation and signal processing. Further, this toolset expands the graphical design for analog–digital computational systems in an open source platform. Figure 5 shows the embodiment of such a tool suite; the representative case of the translation from an application to a heterogeneous set of

analog and digital hardware + software resources.

1.2 Engineering Education

The FPAA is a hardware platform that is capable of realizing both analog and digital systems. Thus, it is not restricted to a digital or analog theoretical course and can also be used for application courses like senior design, Capstone, or independent study. These application classes would most likely use the FPAA in an embedded fashion instead of testing functionality of an unique circuit while changing parameters. Whereas a traditional theory class could use the FPAA for both circuit and system categories by taking a scaffolding approach. A classic analog class would allow for the full range, circuit to system, of topics to be discussed in the classroom. The systems concept would lend itself to a short course or workshop for time constrained events. Thus, the aptitude of the FPAA analog resources can be tested at intervals of increasing complexity.

1.2.1 Methods to Structure a Course

The learning environment of a classroom can take on many forms such as lecture, laboratory, flipped-classroom, blended, and only online. Figure 6 shows different technologies that can be incorporated into the classroom. FPAA SoCs and RASP Tools are applicable in each scenario due to their portable nature. In lecture, the professor has the freedom to have students work individually and in groups with the platform before or after introducing the theory behind the topic of the day. Teachers that prefer different combinations of discovery and practice to solidify knowledge would both be satisfied. Professors would not necessarily have to worry about learning a complicated platform as documentation, instructional videos, and examples will be provided. Laboratory scenarios would give registered students the opportunity to build and confirm their designs with or without an instructor, but at least a teaching assistant for guidance. A flipped class gives students the leisure to watch the supplied

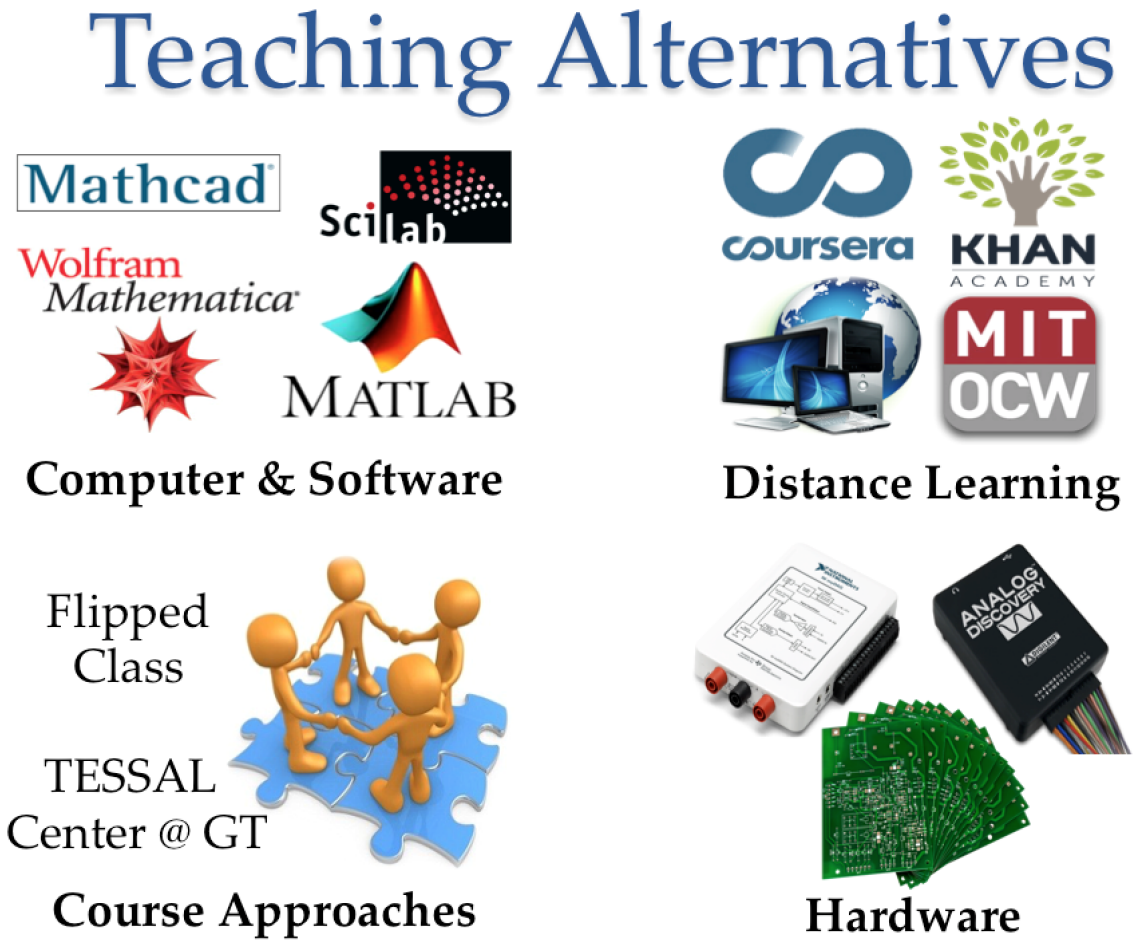


Figure 6: Technology in the classroom whether using software, hardware, or both are used by educators to supplement delivery of course topics. Computer assisted learning environments permit students to solve higher order circuits and non-linear systems. Remote learners can have an equivalent mixed-signal design lab setting with access to CAD tools. Portable and cost effective labs allow multiple resources to be available to numerous students at a time. Pre-fabricated boards and data acquisition apparatuses give students access to experiment with tangible working systems to obtain measurements.

videos anywhere and then have a designated time to come to school ready with questions and eager to tackle problems in the safety of a supportive environment. Lastly, an online course (*e.g.*, massive open online course (MOOC)) that accommodates individuals that want to learn, but due to personal circumstances whether financial,

time restraints, *etc* cannot attend a physical class can experience the same participation and interaction with hardware as students on campus. A possible drawback to MOOCs is limited access to the instructor depending on if any recordings are live and if on-air questions are allowed.

1.2.2 FPAAs and Courses

Georgia Tech has utilized FPAA devices in a classroom setting as a way of innovating teaching techniques for the past ten years [17,36,38,79,81]. We have mapped learning objectives of an analog course to hands-on experiments to be completed either in class or outside of class. Analog courses with laboratories typically have students obtain data with standalone ICs, wires, passive elements, and solderless breadboards that can become convoluted. A single wire connection not securely placed in the breadboard will make debugging cumbersome for large designs. On the contrary, using programmable hardware and tools enable students to intently concentrate on understanding concepts to gain knowledge without being distracted by hardware setup inconveniences. FPAA SoCs are designed to allow students to focus on the circuit operation of analog, digital, and mixed-signal circuits. The advantage to abstract hardware intricacies from students was a dominant benefit of RASP Tools developed for FPAAs. Thus, the potential unkempt hassle of tangled wires was alleviated and the importance of parameter manipulation during phases of design and test was highlighted. With a palette of blocks available and the flexibility to create new blocks, students with varying degrees of comprehension of course topics can be accommodated. We recently decided to incorporate assessment into our pedagogic strategy to measure the effectiveness of our teaching methodology and technology. For this evaluation, students used the FPAA SoC software suite, RASP Tools, and RASP 3.0a SoC boards in the course ECE 6435: *Neuromorphic Analog VLSI Circuits* to promote experiential learning. The culmination of refining our efforts will be a blueprint for

implementing this technique elsewhere.

1.3 Overview of Research

In Chapter 2, I will introduce an analog-digital hardware-software co-design environment for simulating and programming configurable systems. Currently, these systems as discussed in Section 1.1.1.2 are FPAA SoCs that seamlessly combine tiles of analog and digital components using FGs, but can easily accommodate other ICs. The environment, an open source platform, is an unified tool suite that is comprised of numerous custom algorithms, scripts, and other open source software. This program, RASP Tools, enables low-power analog-digital co-design by allowing users to harness the benefits of FGs operating in their multiple roles within the FPAA SoC. Several examples of mixed-signal circuits and systems showcase the capabilities of the toolset in conjunction with the FPAA. These designs utilize the general purpose I/Os, integrated processor, peripherals, DACs, and ADCs.

In Chapter 3, I will talk about the concept of using FPAAs in the classroom. In particular, the focus is the use of hardware and software described in Chapter 2 in an analog course. The wide range of uses in others courses is acknowledged. I delve into the opportunity and flexibility that FPAAs provide instructors in augmenting and updating their pedagogic techniques. The hypothesis that providing students hands on experience is valuable to learning is discussed. The preparation for and improving of the FPAA's toolset and making a PCB for the RASP 3.0a FPAA SoC for a class is also described.

In Chapter 4, I will detail the approach to assess students that used FPAA SoCs and RASP Tools in a graduate level course. The results of an administered pre- and post-survey given to students to obtain their opinion are shown. The questions asked students to rate their prior knowledge and skills, the course teaching methodology impact, and the hardware and software overall. Projects that students completed

during the course are highlighted. Additionally, the other techniques used to assess the class throughout the semester are mentioned.

In Chapter 5, I will conclude by summarizing the research completed. My individual contributions are itemized within this chapter as well. There are five appendices following this chapter that provide more explanation of some topics; the first discusses ode simulation of FPAA blocks in Xcos, the second showcases the pseudo code of algorithms developed, the third displays the RASP Tools startup guide, the fourth depicts the schematics of the RASP 3.0a SoC PCB, and the fifth illustrates the pin layout of the RASP 3.0a SoC and RASP 3.0 SoC PCBs.

CHAPTER II

FPAA SOC TOOL INFRASTRUCTURE

Large-scale mixed-mode configurable systems, namely FPAA SoCs [32], reveal the demand for design automation tools. These tools permit users to proactively design, iterate, and discover evidence of solutions for the extensive list of open questions within an analog–digital co-design space. This chapter presents a unified tool framework for analog-digital hardware-software co-design [18]. Users of this tool infrastructure can contemplate design choices (*i.e.*, power, area) when considering mixed-signal computation and signal processing endeavors. This open source tool suite, RASP Tools, indicates an inaugural time for the analog–digital software co-design discussion. Further development through open source platforms can refer to this venture when future mixed-mode configurable systems become available.

RASP Tools integrates a high-level design platform built in Scilab/Xcos [71] (open source alternatives to MATLAB and Simulink, respectively) with the compilation tool **x2c** (Xcos to chip) to create a design environment that interacts with configurable and programmable hardware. Figure 7 illustrates using this tool framework to compile down a Xcos design to a programmable system through **x2c**. Thus, **x2c** provides a time-efficient method of solution that affords system designers the ability to integrate useful components. It also empowers circuit experts to frequently develop creative and reusable circuits and systems within the same design environment. This tool platform unites the existing open source Versatile Place and Route (VPR) tool [54] with custom software to develop an integrated environment to simulate and experimentally test designs on FPAA SoCs.

High-level software built in Scilab/Xcos converts user created circuit and system

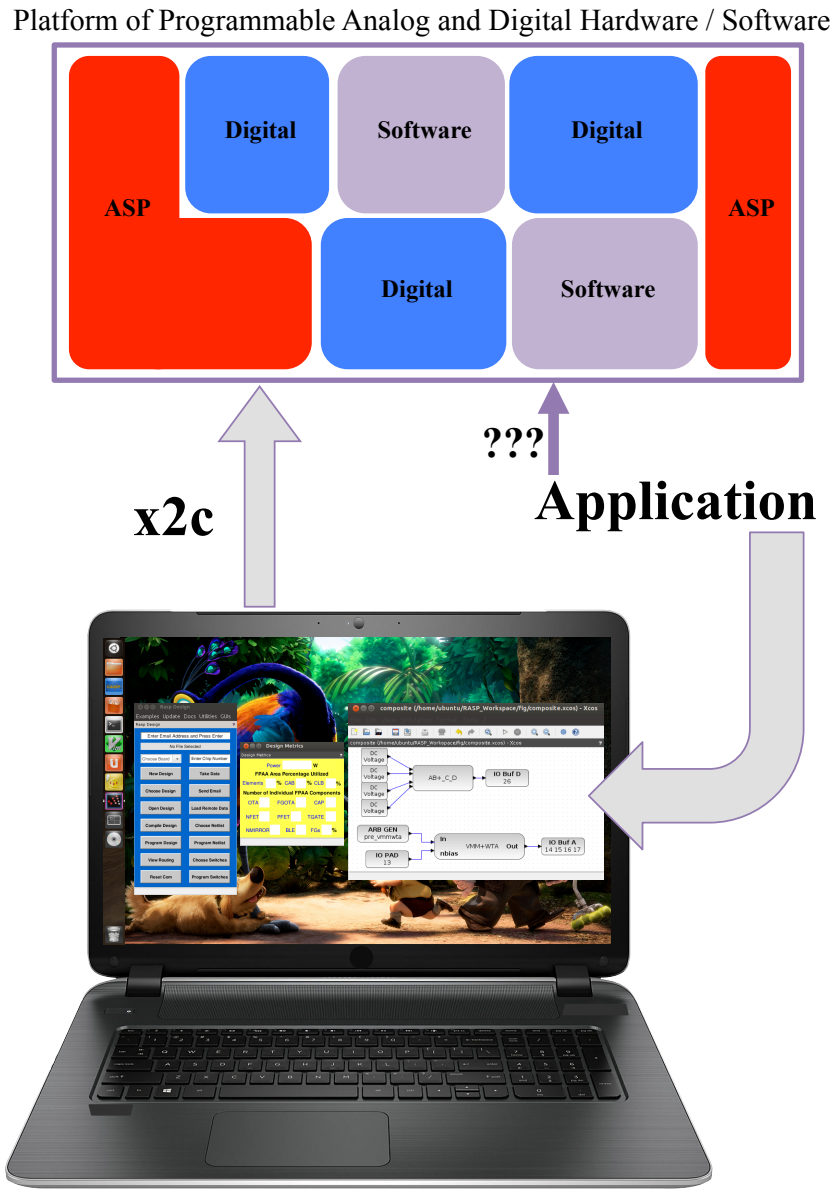


Figure 7: The translation from an application to a heterogeneous set of digital hardware + software resources is a known field of study; in contrast, the same scenario for analog and digital hardware + software assets is a question that is barely even considered. Software tools that encapsulate a range of potential application solutions were written in the Scilab/Xcos environment to enable a range of system design choices to be investigated by the designer. The software infrastructure developed permits high-level simulation and compilation to physical hardware through the **x2c** tool. This design automation tool, RASP Tools, is publicly available at http://users.ece.gatech.edu/phasler/FPAAtool/index.html

designs to BLIF (Berkeley Logic Interchange Format). The BLIF file generated by the **sci2blif** (Scilab to BLIF) tool is one of the inputs given to the modified VPR

tool, *vpr2swcs* (VPR to switches). Then *vpr2swcs* encodes input information with the personalized architecture files of the user selected programmable chip platform, which results in a targetable switch list for the configurable analog–digital system. The abstracted blocks in the user's design are acquired from tailored Xcos library palettes that contain various analog and mixed-signal components. Therefore, existing users and future researchers can easily access these basic analog operations and computations.

Section 2.1 overviews the analog–digital design tool. Section 2.2 describes tool integration with an experimental FPAA platform. Section 2.3 describes the methodology for implementing the toolset, including macromodel system simulation consistent with measurements and translation of a Xcos diagram to a netlist for hardware compilation. Section 2.4 demonstrates the approach to determine macromodel equations that correlate with measurements of an example system. Section 2.5 summarizes this chapter.

2.1 Analog-Digital Design Tool Overview

Figure 8 depicts the tool flow block diagram that enables experimental IC testing. The Xcos environment was designed to enable macromodel simulation of a physical system. The **x2c** tool converts a Xcos design to a switch list that is used to program the hardware; it is comprised of **sci2blif**, which converts Xcos to BLIF, and *vpr2swcs* that converts BLIF to a programmable switch list. The tool *vpr2swcs* contains wrapper code to augment the open source VPR tool [54], a tool originally designed for placing and routing circuits in FPGA architectures. The particular system to be targeted, an FPAA IC, is defined by its own technology file for **x2c** tool use.

Figure 9 portrays a full tool example of the graphical interface and both the simulation and experimental results for a first-order low-pass filter (LPF) system. The tool is encapsulated in a single open source Ubuntu Virtual Machine with a

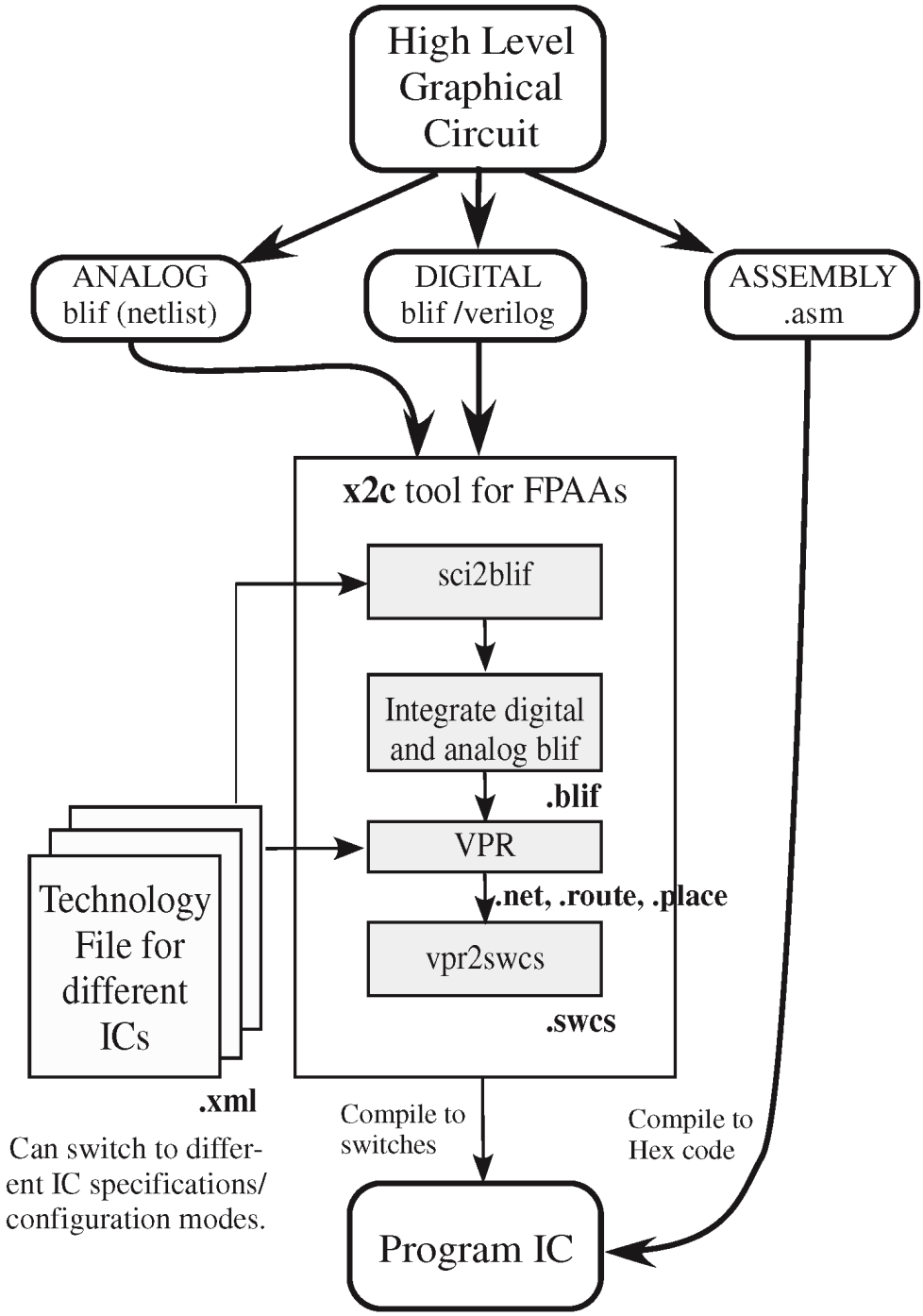


Figure 8: Top-down design tool flow. The graphical high-level tool has a library of palettes that contain available blocks that compile down to a combination of digital and analog hardware devices, as well as software (processor) resources. The **x2c** tool converts a Xcos design to a switch list for programing the FPAA SoC. It combines open source software such as Scilab, Xcos, Virtual Place and Route (VPR), and custom algorithms (**sci2blif** and *vpr2swcs*) into a software suite to program and test FPAA SoCs.

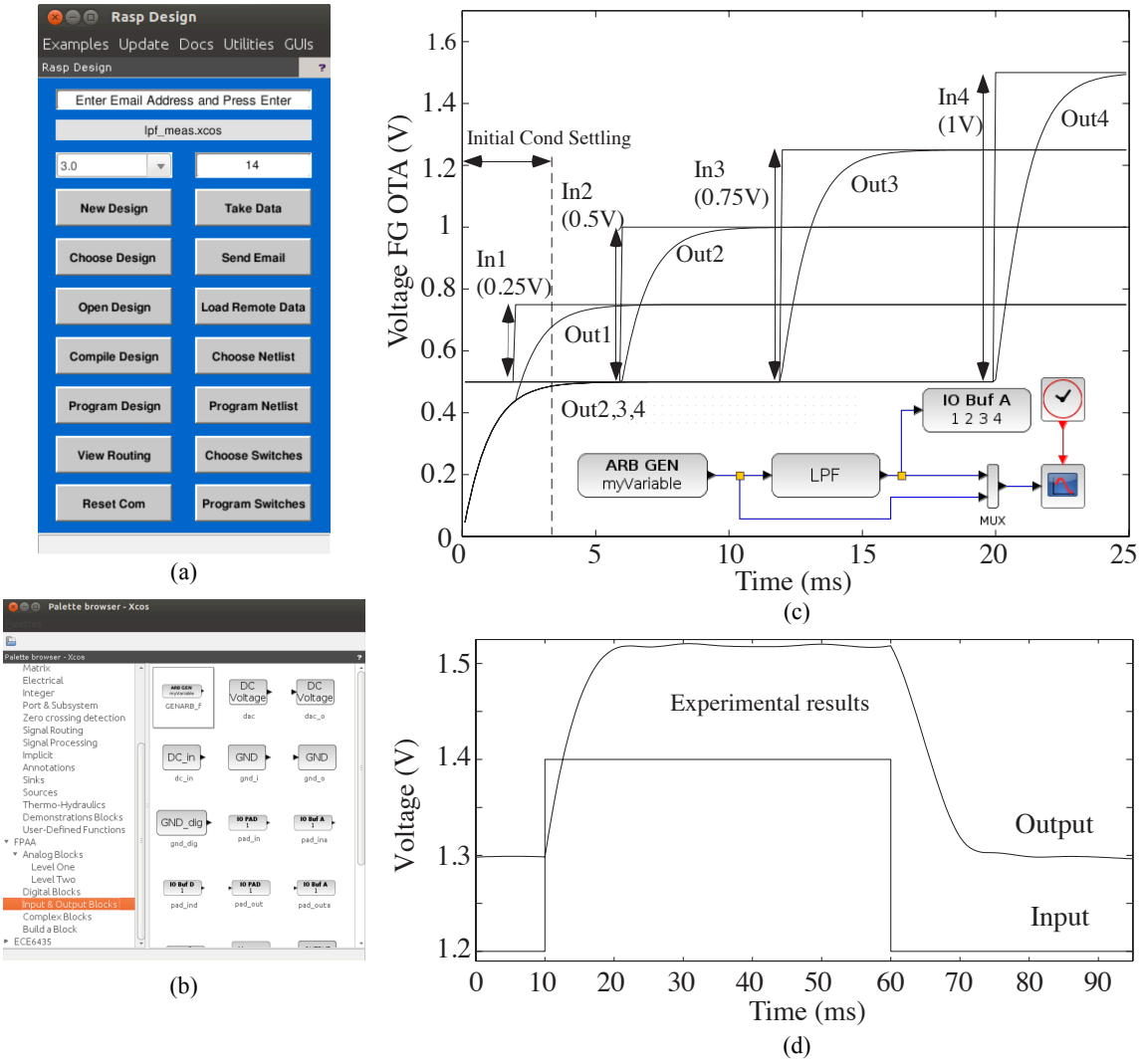


Figure 9: An example of the entire tool flow for a low-pass filter (LPF) computation. (a) The user chooses basic design options through the RASP Tools GUI, which starts running when the Scilab tools are initiated in the distributed Ubuntu Virtual Machine (VM). (b) Snapshot of the Xcos palette for FPAA blocks. There are four sections, namely analog, digital, input/output, and complex blocks. The analog, digital, and I/O blocks represent the members of the different tiles in a chip. Complex blocks are pre-defined circuit blocks that utilize more than one of these elements. (c) Simulation results for a four input/output computation. Connecting lines and blocks allow for vectorized as well as scalar inputs. Inset shows the Xcos diagram where the user sets parameters for simulation or for compiling into an integrated circuit. (d) Experimental results for a one input/output computation.

single desktop button to launch the entire Scilab tool framework. Xcos gives the user the ability to create, model, and simulate analog and digital designs. The Xcos editor has standard blocks that are compartmentalized into classes or palettes that

range from mathematical operations to digital signal processing. The editor allows the internal simulator to utilize the functionality of each block to compute the final answer.

The Xcos tool supports user-defined blocks and libraries [71]. When the user opens the Xcos editor, a palette browser is displayed, as shown in Fig. 9(b). The browser lists Scilab's collection of palettes as well as user-defined palettes. One selects from a palette of blocks to build a system, which can be composed of a mixture of analog (BLIF), digital (Verilog), and software (assembly language) components. The palette for RASP Tools contains sub-palettes for blocks that are classified as analog, digital, inputs/outputs, and complex blocks. The input/output palette contains ordinary circuits like DACs, an arbitrary waveform generator, I/O pads, and voltage measurement devices (ADCs). The digital palette contains typical circuits like a digital flip-flop and a clock divider. A few examples of complex blocks are a low-pass filter (LPF), a sigma-delta ADC, and a Vector Matrix Multiplier (VMM) connected to a Winner-Take-All (WTA) block as a classifier structure.

Block characteristics are stored in a Scilab data structure that is populated by a block's two configuration files, an interfacing function and a computational function [See Appendix A]. The interfacing function defines the dialog box that retrieves and stores values for a block in the Xcos framework. These dialog boxes have adjustable parameter fields and default values can be specified. The interfacing function also defines the size of the block and its number of input ports and output ports. Consistency checks are included to let users know if their entered values are valid.

Each block as mentioned has two files that dictate its appearance and performance within Xcos. The computational function encourages model customization, while the interfacing function supports built-in data checking, variable inputs/outputs, and default parameters. The interfacing and computational functions are heavily coupled by the Scilab structure for a block. Thus, the parameters retrieved from a block's

dialog box are accessible to the computational function to compute the output of a block during simulation.

The following discussion ensues a previous representation [67] of analog blocks that are either considered Level 1 or Level 2. Level 1 blocks abstract away intricacies for system designers, such as inputs and outputs being vectorized and voltage-mode signals (as seen in Fig. 9(c) – the bus of four signals). Figure 9(d) shows experimental results for a single line (bus of one signal). Level 2 blocks allow for general circuit design, usually a representation of CAB elements. Example CAB elements include transistors, floating gate (FG) transistors, transconductance amplifiers (TA), current mirrors, and digital transmission gates (T-gate). The TA output current is proportional to its applied input voltages for one operating region. A T-gate emulates a rail-to-rail switch element in a CMOS process. Each block uses vector signals and vector-based block computation. Each block may represent potentially N virtual blocks (or more) to either be compiled to silicon or simulated. The lines or links that connect the blocks together in a Xcos design are essentially layered buses. Each link and connected block allow vectorized signals to pass, as seen by the example in Fig. 9(c) (four signals on one line for simulation). This occurrence is consistent with the Level 1 definition [67] that empowers users to simply develop systems without extra clutter.

2.2 Integrating the Analog–Digital Design Tool with an FPAA Platform

The test platform is a full system IC requiring only a simple interface to the outside world through a USB port, which most times is a standard peripheral to a normal device. Figure 4 shows the block diagram of the FPAA SoC used in this chapter for experimental measurements. This FPAA SoC enables nonvolatile digital and analog programmability through FG devices, both in the routing fabric as well as in CAB block parameters (*i.e.*, bias current for an OTA). The tiles with digital (D)

components are CLBs and the tiles with analog (A) and digital components are CABs. Further, the use of FG devices for switches effectively embeds analog components into the routing fabric in addition to permitting connections on these lines [80]. As a result, far more computation can be done in the routing fabric when compared to the CAB or CLB tiles. Any tool development for these FPAA SoCs must be able to handle these discussed attributes; almost all configurable systems will have similar characteristics that must be encoded in the system's technology file repository. Other FPAA devices fit into this general framework [7, 8, 21, 46, 66, 85].

Figure 10 illustrates descriptions of blocks found in the tool's library palette (all Level 1 blocks) and their related circuit schematics; specifically in this figure are filters, counters, and peak detectors. Although a circuit expert gains tremendous insight by tinkering with a particular circuit to be compiled and programmed on the IC, most system designers are satisfied with getting the desired functional behavior with minimal non-idealities from the circuit. The result is a rich set of analog and digital blocks, similar to FPGAs when using graphical design tools (*i.e.*, [57]), that can be expanded and enhanced as desired. The DC voltage block in Fig. 14 and Fig. 15 is a FG programmed transconductance amplifier that provides a low-impedance DC voltage output that is consistent with the voltage value provided in the Xcos design.

Figure 10 also details common CAB components and a classic routing infrastructure of input/output lines connected to a Manhattan geometry routing fabric; the detailed routing to compile a C^4 band pass filter circuit is exhibited. A macroblock is an abstracted complex circuit block constructed from the elements within a single CAB or CLB. As a result, the macroblock's pre-optimized internal routing gives VPR the task to only place and globally route the block. Macroblocking symbolizes to a extent one objective of a circuit designer, which is to begin with a Level 2 block and create an equivalent Level 1 block.

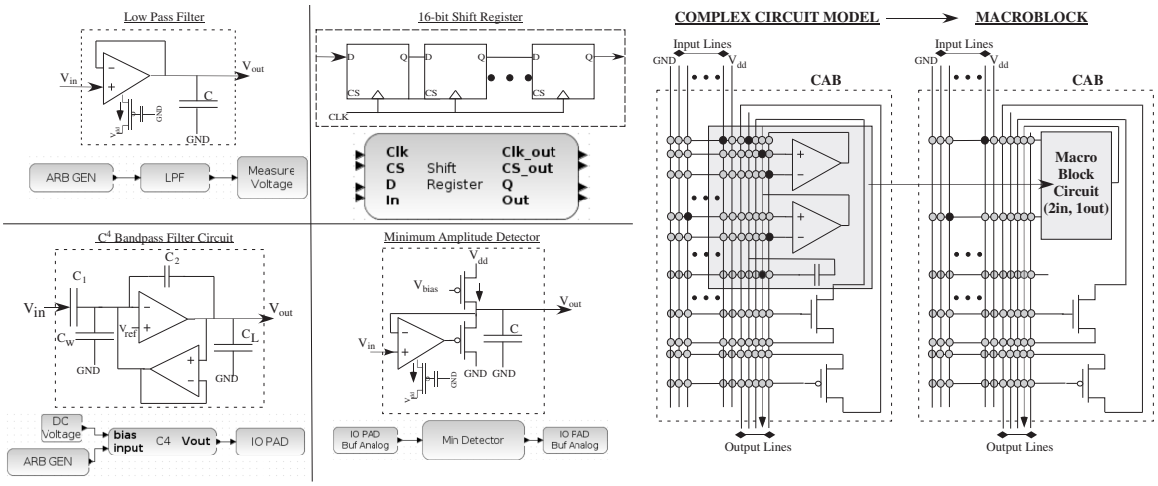


Figure 10: An example of a range of low-level circuit components and their block diagram as well as some of their testing circuits, which include analog and digital components. We show a LPF (as seen in the previous example), a minimum amplitude detector, a capacitively-coupled current conveyer (C^4) band pass filter, and a digital shift-register block. We are able to draw block diagrams for all mixed-mode computation; in each of these cases, the inputs could be a scalar or a vector. Often a user will want to encapsulate the knowledge as much as possible from a working design. For example, an analog designer might want a series of circuit devices to be grouped into a single CAB; a macroblock takes indicated elements of analog and digital tiles and restricts them to reside within a single CAB or CLB. Separate black-boxes in VPR are used to categorize and maintain all macroblocks.

2.3 Methodology for Implementing the Tool Suite

This section communicates the key aspects of the RASP Tools infrastructure. Figure 9 displays a single Xcos block diagram structure used for both simulating macromodels and compiling designs to hardware. This section follows the Level 1 definition [67], as defined elsewhere and first fully implemented in this work.

For Level 2 cases, the compilation to BLIF from Xcos follows the same path as Level 1. However, the simulation environment commands a far more complex simulation domain. Simulation for Level 2 requires compiling a netlist into a SPICE model and next either taking a direct measurement or executing a robust simulation using Scilab's Modelica modeling syntax.

The following sections address the stipulations for Level 1 macromodeled simulation and the aspects required for **sci2blif**, which converts the Scilab structure into a

format ready for place and route compilation.

2.3.1 Macromodel Simulation

A typical design flow includes simulating a design's functionally, analyzing the results, and iterating to a good solution before proceeding to hardware synthesis. This process is often a result of constraints preventing access to a hardware system. If physical hardware is directly available (and portable), one might question the decision to not always directly take measurements and get precise results in real time. Even in cases where hardware is available, it is often useful to have one simulation case for testing DC values that provides reference simulated data to compare with experimental measurements. The amount of simulation one might do before compiling a circuit will depend on compile time (longer compile time, more simulation), accessibility to FPAA hardware, whether hardware is local or remote, user inexperience (more inexperienced, longer simulation time) as well as the number of potential debugging points.

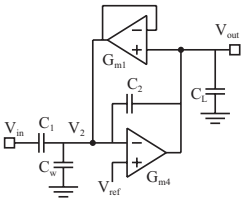
The simulation approach we use focuses on an as fast as possible simulation model that gives accurate results, unlike generalized SPICE models that include all transistor configurations for numerous situations. A given macromodel has precisely one specific case related to a particular hardware device, greatly simplifying the resulting computations. The final simulation and experimental results should be reasonably close (*i.e.*, within 1%–5%), due to this sufficient computational complexity formulation. Scilab, like MATLAB, optimizes for vector operations; vectorization of blocks preserves this functionality and achieves the fastest possible numerical simulation.

The analog modeling system requires the use of ordinary differential equations (ODE), potentially in combination with algebraic equations that capture the continuous-time circuit nonlinearities. Scilab also enables discrete time modeling as well as modeling for clocked systems in a similar manner. In the FPAA SoC, every node or

connection point will have an associated capacitance. These capacitances exhibit dynamics that are embedded in state variables of ODEs. The required format for a Xcos ODE simulation is of the standard form:

$$\frac{d\mathbf{V}}{dt} = \mathbf{f}(\mathbf{V}, \mathbf{V}_{in}) \tag{1}$$

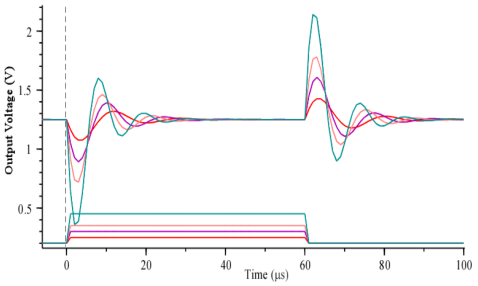
where $\mathbf{V}$ is the vector of state variables (*i.e.*, voltages) and $\mathbf{V}_{in}$ is the vector of system inputs. The final ODE definition is put into the computational function code of a block using this functional form.



(a)

Modify ODEs of the form
$\frac{d\mathbf{V}}{dt} = \mathbf{f}(\mathbf{V}, \mathbf{V}_{in}, \frac{d\mathbf{V}_{in}}{dt})$

Modify ODEs to have
no input derivatives
$\frac{d\mathbf{V}}{dt} = \mathbf{f}(\mathbf{V}, \mathbf{V}_{in})$



(b)

$$(C_1 + C_2 + C_w)\frac{dV_1}{dt} = C_1\frac{dV_{in}}{dt} + C_2\frac{dV_{out}}{dt} + I_{bias2}\tanh\left(\frac{V_{out}-V1}{V_L}\right)$$
$$(C_L + C_2)\frac{dV_{out}}{dt} = C_2\frac{dV_1}{dt} + I_{bias1}\tanh\left(-\frac{\kappa V_1}{2U_T}\right)$$

(c)

$$C_{eq}\frac{dV_1}{dt} = C_1\frac{C_L+C_2}{C_2}\frac{dV_{in}}{dt} + \frac{C_L+C_2}{C_2}I_2 + I_1$$
$$C_{eq}\frac{dV_{out}}{dt} = C_1\frac{dV_{in}}{dt} \frac{C_1+C_2+C_w}{C_2}I_1 + I_2$$
$$I_2 = I_{bias2}\tanh\left(\frac{V_{out}-V1}{V_L}\right)$$
$$I_1 = I_{bias1}\tanh\left(-\frac{\kappa V_1}{2U_T}\right)$$

(d)

$$C_{eq}\frac{dV_1}{dt} = \frac{C_L+C_2}{C_2}I_2 + I_1$$
$$C_{eq}\frac{dV_{out}}{dt} = I_2 + \frac{C_1+C_2+C_w}{C_2}I_1$$
$$I_2 = I_{bias2}\tanh\left(\frac{V_{outa}-V2+(\beta-\alpha)V_{in}}{V_L}\right)$$
$$I_1 = I_{bias1}\tanh\left(-\frac{\kappa(V_2+\alpha V_{in})}{2U_T}\right)$$

(e)

Figure 11: Approach to building a Level 1 macromodel for the C^4 filter that correlates to measured experimental data. (a) Circuit diagram for a C^4 band pass filter. (b) Simulation of a step response for the C^4 band pass filter. (c) Starting equations for the circuit in (a). (d) Modification of the equations into the 1st form. (e) Modification of the equations into the final Xcos ODE form.

Figure 11 shows the steps to formulate a physically realistic model for a C^4 band pass filter (BPF). A particular system will require reformatting these vectors from typical circuit analysis. The example C^4 band pass filter is modeled at the circuit function level, not just the linear transfer function level shown elsewhere. Nonlinearities are modeled accurately (as seen by the *tanh()* function) to enable a system

designer to minimize the effect where needed as well as equip a designer to utilize nonlinearities when desired. Figure 11 shows three mathematical iterations required to transform classically written circuit equations into their proper Xcos simulation form as well as Xcos simulation data for this model; the simulation data corresponds closely to the experimental data.

2.3.2 sci2blif: Xcos → VPR

When the user presses *Compile Design* on the graphical interface, Scilab invokes **sci2blif** [See Appendix B Algorithm 1]. The **sci2blif** tool translates a circuit created in Xcos to BLIF for *vpr2swcs* to generate a switch list; assembly language modules are also gathered during the process. Analog blocks are converted through **sci2blif** into a BLIF format. Digital blocks use a fraction of the Verilog-to-Routing (VTR) tool [54] to transform Verilog into BLIF statements. The switch list represents the low-level hardware description (*i.e.*, switches to be programmed).

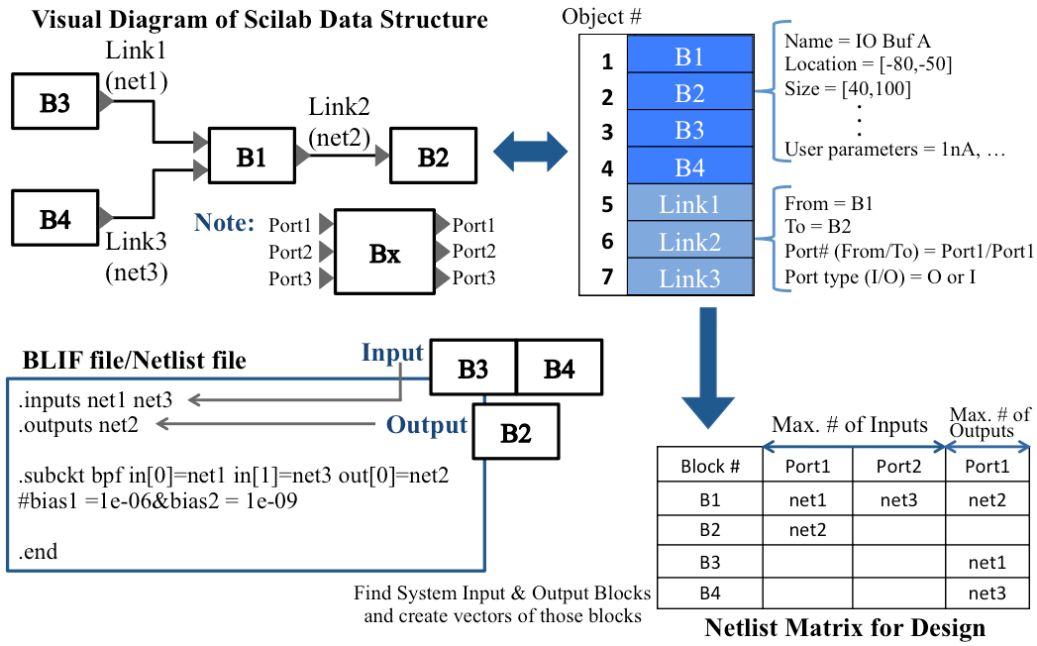


Figure 12: **sci2blif** fundamentals: Compilation of Xcos model to BLIF netlist, which is given as an input to *vpr2swcs*. The data structure, *scs_m*, for a single Xcos diagram is an array with the block information as well as link information. Blocks and links are enumerated as they are populated in the Xcos design. This data structure is used to generate a BLIF definition for VPR.

Figure 12 illustrates the conversion from a Xcos visual representation to BLIF for the analog components; the digital procedures are similar, although typically simpler. Scilab saves the netlist graph that describes the Xcos file contents in its data structure, *scs_m*, as shown in Fig. 12. The block objects are listed first and then the link objects follow in numerical order.

The BLIF file is completed with three passes over the *scs_m* data structure. The first pass parses the *data* block portion to determine the number of blocks that will be compiled to CABs, CLBs, input blocks, and output blocks. The input and output block object numbers are saved in two separate vectors as shown in Fig. 12. B is the number of blocks; I is the maximum number of inputs; O is the maximum number of outputs. Finally, a netlist matrix, G, of size $[B \times (1 + I + O)]$ is generated to contain the net-name numbers that correspond to each block's input and output ports.

The second pass parses link *data* to populate the netlist matrix [See Appendix B Algorithm 2]. The link *data* states whether a block's input or output port is connected to another block's input or output port. Each link is represented by two values: the source and destination. The information retrieved is the block number, port number (ports on blocks are numbered top-down for inputs and outputs), and whether the port is an input or output. These details dictate the index within the netlist matrix to place the net-name number. Figure 13 shows that when users connect an output of a block to at least two inputs, an extra small block is automatically inserted into the Xcos diagram; as a result, these blocks are essentially discarded to reflect the actual connection between blocks as the user intended. In addition to maintaining the netlist matrix, local routing capacitance for each block is also saved in another matrix structure to be used in the third pass.

The third pass parses block *data* to generate BLIF statements for compilation. The input and output vectors as well as the netlist matrix are used to place the net names of inputs and outputs at the beginning of the BLIF file. Then the command

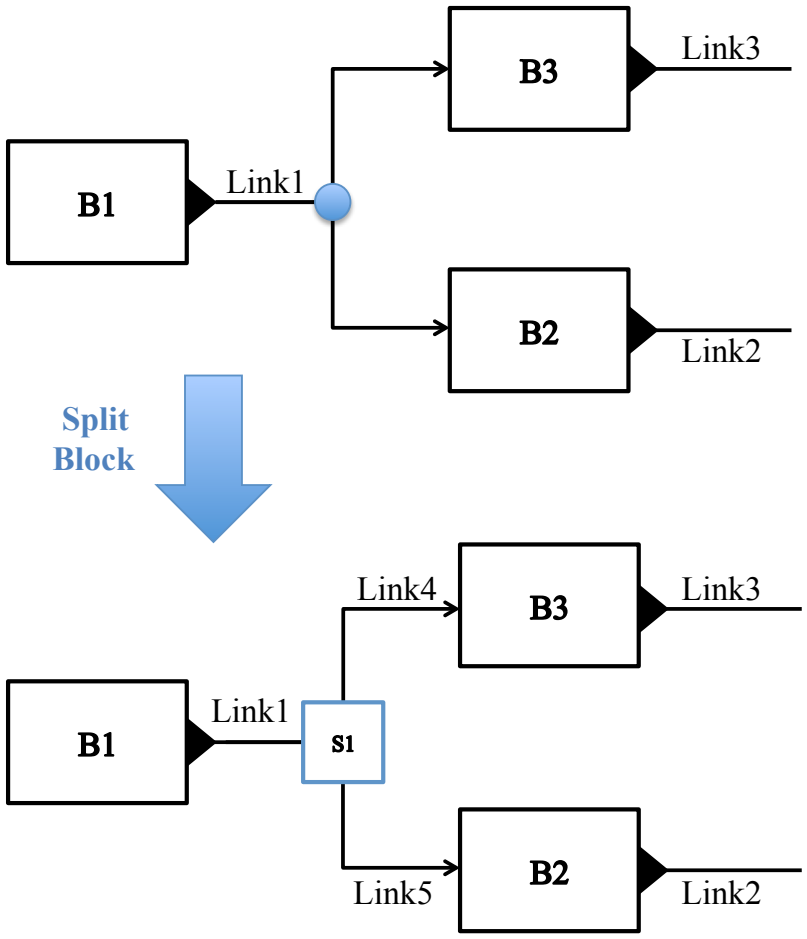


Figure 13: The data structure supporting the Xcos environment only allows a particular link to have a single input and output delineation; therefore, Xcos includes additional blocks (split blocks) to assist the user's objective in connecting a single output to multiple inputs.

for each block is identified, where the net-name numbers are retrieved from the netlist matrix using the block's id number. The command for a block if needed also contains instructions to use routing capacitance to determine Ibias values to be included in the BLIF file (Ibias values are used for programming FGs). This capacitance takes into account global and local routing. The global routing capacitance is determined during a separate process [See Appendix B Algorithm 4 and 5].

2.3.3 Integrating the μP Toolflow

This FPAA IC presents the opportunity of integrating μP code with the configurable analog and digital fabric. The challenge of design partitioning as well as integrating assembly code blocks into the macromodeling simulation for an implemented system arises. For example, the arbitrary waveform generator block in Fig. 10, Fig. 14, and Fig. 15 is an example of a block with high assembly language code content. The voltage output of the block goes through one of multiple signal DACs on the targeted IC. Processor memory space is allocated for the vector of input voltages defined in Scilab for this operation; assembly code is generated and integrated into the final FPAA programming flow. A similar block for recording data uses one of multiple ADCs that are available either directly on-chip or compiled on-chip.

One builds assembly blocks similar to analog or digital FPAA fabric blocks. The designer would write and test the associated assembly code block and abstract its functionality. These resulting digital blocks are Level 1. The block designer would also write the accompanying macromodel simulation code. These simulations typically utilize the discrete-time simulation modeling available in Xcos.

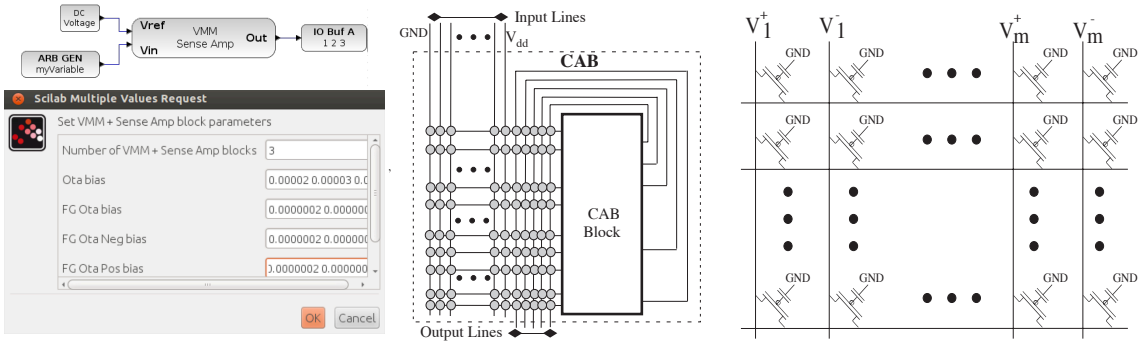


Figure 14: A key feature of these tools is the ability to build useful computation out of routing resources. For example, the synthesis, placement, and routing of circuits containing VMMs are built out of routing (*i.e.*, floating-gate) switches. From left to right, block diagram and parameters for a VMM block, VMM built from a crossbar switch matrix, and local interconnect routing resources inside an analog tile.

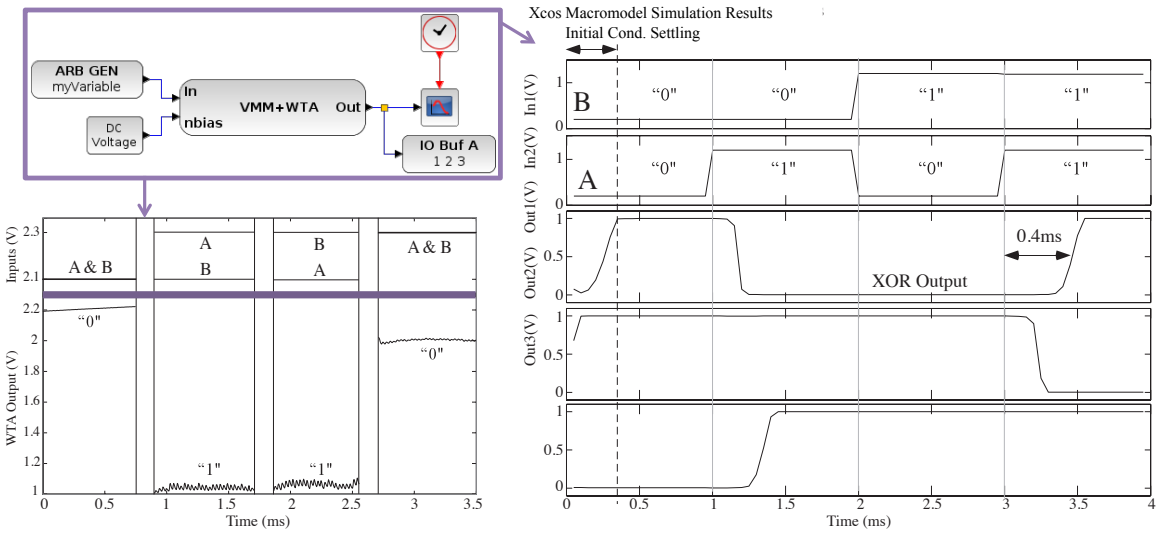


Figure 15: A system example showing a simple circuit classifier built from a three-input $\times$ three-output VMM + WTA. The Xcos circuit diagram, with its vectorized connections, can be simulated through Xcos macromodel simulation as well as compiled and measured experimentally on an FPAA IC. The three-input vectorized system, a 3×3 VMM, is configured as a two-input XOR gate (the other input is fixed). The functional and dynamic behavior agrees well although the signals have different DC offsets.

The challenge in general of establishing a generic set of digital blocks is developing a nano-size operating system (100–300 bytes) on the processor. Minimizing the reliance on large on-chip memory size is critical because the microprocessor in this IC (open source MSP430 μP [59]) is eight times smaller than the 16k (16-bit word) memory bank (measured and designed across processes from 350nm – 40nm). One sees that large memory space is highly expensive and power hungry for embedded applications. This design tool interface provides developer-friendly system design without requiring a large embedded system memory; the primary reason for larger operating systems is to support developer creativity. The tool also requires being able to estimate the memory required for all blocks and making that information available to the user.

2.4 Simulation and Experimental Data from the Compilation Tool Example

This section details a system example and highlights the use of routing components as computational elements. The work in [32] shows a number of further examples compiled using RASP Tools. The two examples described below purposely illustrate the tool suite approach and not the capability of the particular FPAA device. Currently, we have tested a particular SoC device to operate at a 1-MHz frequency where macromodeled simulation reaches similar levels without any further issues as expected.

A crossbar array of analog programmed FG switches can compute an analog VMM, a common operation in signal processing, entirely in routing fabric. The inclusion of such capabilities require additional sophistication at all levels of the tool flow. Figure 14 depicts a Xcos vectorized block diagram that implements a VMM structure, its implementation using a local routing crossbar array, and a circuit representation for this VMM computation. The FPAA [32] uses FG switches to enable analog computing when programming a switch to an analog value. This circuit requires a current-to-voltage SoC conversion, in this case a transimpedance amplifier of two OTA devices is consistent with Level 1 requirements. The resulting block has a dialog box to obtain key parameters.

The VMM + WTA classifier elegantly compiles into FPAA ICs using the VMM consisting of FG routing fabric devices. This classifier with a k-WTA topology is a universal approximator represented in a single layer [64]. Figure 15 illustrates a VMM + WTA circuit, used as a classifier circuit, where the VMM block is a single vectorized block.

Figure 15 showcases VMM + WTA macromodeled simulation data and measurement data. The vectorized block models this circuit as three (vectorized) key equations. These macromodel equations are derived from sub-threshold transistor equations with realistic approximations, such as some transistors in saturation, that were experimentally verified. The VMM computation, based on FG devices, with outputs going into cascaded WTA inputs, is modeled:

$$\tau_1 \frac{d\mathbf{y}}{dt} + \mathbf{y} = 2(1 + a^2 \mathbf{x}^T \mathbf{x}) + \mathbf{W} \sinh(a\mathbf{x})/N \tag{2}$$

The transistor operation models the high-gain and negative feedback aspects of the WTA block and computes with the representation:

$$A_k = e^{\Delta V_{a,k}/U_T}, Z = e^{\Delta V/U_T}$$

where $\Delta V_{a,k}$ are the input gate voltages (around a steady-state bias voltage) for a classic WTA circuit [48] and ΔV is the common-source voltage (around a steady-state bias voltage) for the extended differential pair structure for a classic WTA structure [48]. The (algebraic) modeling equations below follow these definitions:

$$\tau_A \frac{d\mathbf{a}}{dt} = \mathbf{y}.^*\mathbf{a} - Z(\mathbf{a} - r\mathbf{1}) \tag{3}$$

$$\tau_Z \frac{dZ}{dt} = \mathbf{1}^T \mathbf{a} - Z \tag{4}$$

where the vector of ones ($\mathbf{1}$) and $r = e^{-V_{a0,k}/U_T}$ serve as a constant that relates a coefficient to affirmed biasing values around their steady-state point. The output termination uses a typical common-source or common-gate amplifier configuration based on the values of A and Z.

Figure 15 shows experimentally measured VMM + WTA data from the same vectorized test system. The number of vectorized inputs and outputs of a VMM +

WTA block correspond to the size of the classifier determined by the user. A two-input classifier and another fixed input within a 3×3 VMM matrix demonstrates XOR functionality; for WTA circuits, a low output voltage signifies a winner.

2.5 Summary and Approaches for Analog–Digital Co-design

This chapter presented an analog–digital software-hardware co-design environment and mixed-signal design examples using FPAA SoCs. RASP Tools simulates designs as well as enables experimental measurements after compiling to SoCs in the same integrated design tool framework. This tool suite is situated within an open source Ubuntu virtual machine enabling straight-forward user setup as well as inviting contributions from third party users. Thus, a means to empower a wider community to do analog and digital system design is available. Digital co-design questions pose issues for systems of mixed hardware (*i.e.*, FPGA) and software (*i.e.*, code running on processor(s)). The particular partitioning of the intended computational system is based on metrics of power, area, time to market, *etc.* The recent inclusion of programmable and configurable analog computation allows this community, already a vibrant field, to fundamentally revisit these tradeoffs and issues.

The need for large-scale design tools for FPAA SoC devices was the driving force to develop our tool suite. Our approach is entirely extendable to a wide range of analog–digital programmable-configurable systems. Where **x2c** converts high-level block user descriptions to a BLIF file utilizing **sci2blif** (scilab $\rightarrow$ BLIF), *vpr2swcs* (VPR $\rightarrow$ switches), and modified architecture files; the BLIF file is an input to the modified VPR tool [54]. RASP Tools possesses and allows users to build analog as well as mixed-signal components to form a versatile library. Thus, users and future researchers have access to an assortment of analog operations and computations.

The designer has a few tools available to handle analog–digital computation that are capable of block level simulation, particularly of a fast high-level simulation tied to

experimental data, as well as compilation to experimental hardware such as the FPAA SoC IC example used in this writing. Digital FPGA devices have tools that work with Simulink (MATLAB) [2,3,57,86], and this work provides the first step along the journey to enable similar tools for analog–digital computing systems. The alternative approach requires analog and mixed-signal design expertise, often multiple people for a bottom-up design of a particular custom system using Cadence, Mentor Graphics, or similar tools with their associated costs and complexity [12,33]. Similarly, individuals would have to start with a tool to configure a single IC of pre-built fixed components ("proven resources") interconnected by an optional metal layer [25].

Devices with the complexity of the FPAA SoC make having such tools essential, not just nice to have, for system design. As the call for open source FPGA architectures and tools grows stronger (*e.g.*, [53]), this effort includes and, by incorporating analog computation, expands on this original vision. Simply writing simulation code in MATLAB might find a way forward on simulation, but not as far when compiling and programming the actual design to gain confidence that it will work in practice experimentally. System design is rarely constrained by good high-level algorithm ideas, but rather resources that connect the high-level algorithm ideas all the way through hardware; this tool suite enables such a direct design, encoding the wisdom of the hardware designer where possible in the block library.

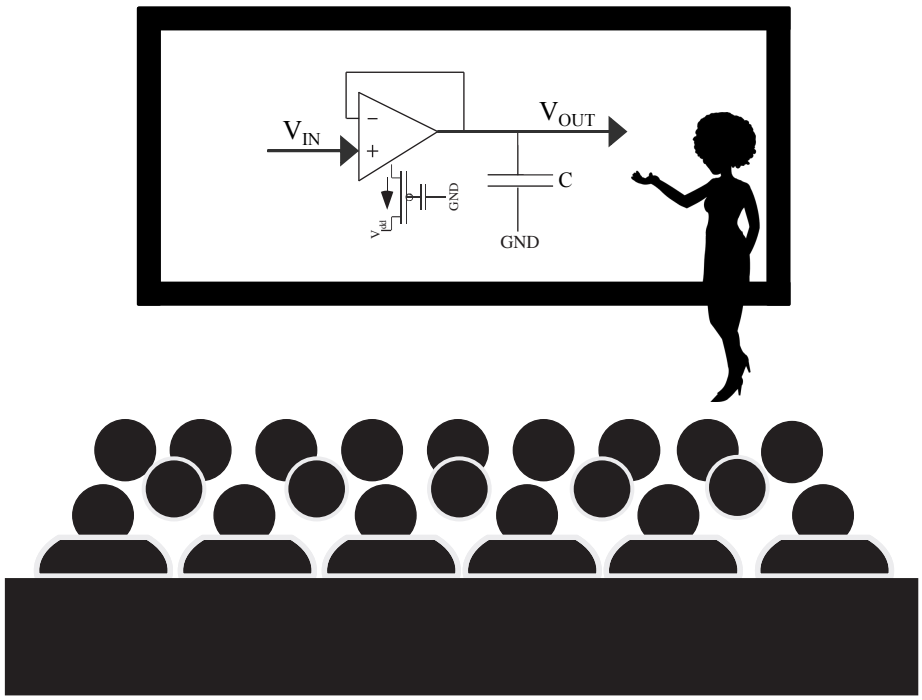
CHAPTER III

FPAA INTEGRATION IN EDUCATION

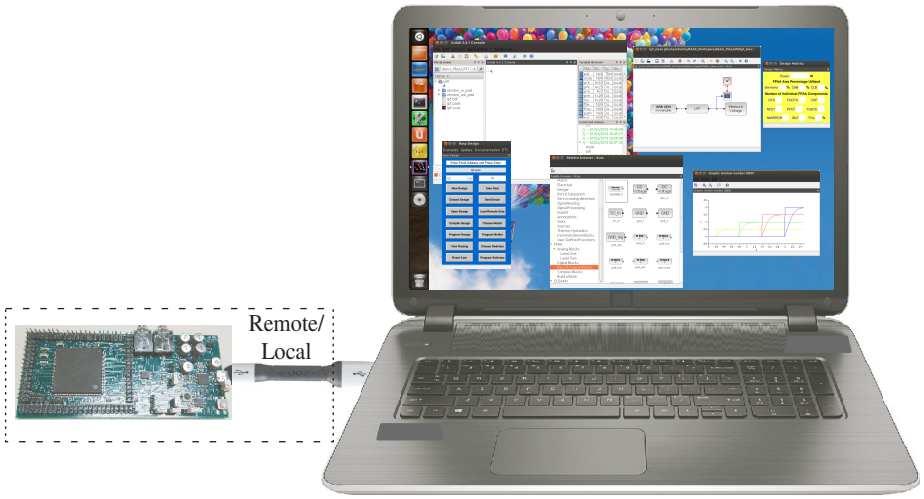
In general, engineering schools with analog design programs do not have silicon hardware resources for students to practice their knowledge of circuit concepts at all levels. Engineering is a field that requires theoretical as well as practical experience; the university is charged with ensuring its graduates are prepared for their future careers. Laboratory exercises are critical in solidifying theoretical and practical confidence in and mastery of our profession [26]. Thus, either having dedicated laboratory sections or incorporating simple lab projects in the lecture itself to achieve this goal are feasible options. Laboratory experiments should be designed to get students excited about the community and thinking about the impact they will make in their career using their past experiences and expertise. There are some analog courses that have a separate laboratory to complement the co-requisite lecture at the undergraduate level as shown in Fig. 16(a). There are other undergraduate courses that have adopted experiment modules to lessen the use of traditional lecture-based courses [5]. Essentially at the undergraduate level this idea has been implemented to a degree, but not so much at the graduate level. The essence of this research is situated within the graduate studies domain.

3.1 System Design in the Classroom

Advances in technology have allowed professors to introduce teaching aids, software, and hardware to supplement lectures in a course to enable design, simulation, and testing of circuits. For analog system design classes this is a challenge. Typically, laboratory projects are assigned to allow students the opportunity to gain hands-on simulation experience outside of lecture, but not to obtain data from fabricated



(a)



(b)

Figure 16: The analog system design classroom. (a) Here we illustrate a typical analog system design classroom environment, where laboratory is a separate component for testing hardware. (b) Approach proposed with FPAAs, where the students can conduct experiments in the classroom using their laptops and an optional FPAA board. The students use our open source software, RASP Tools, to program FPAA SoC boards. A board is either connected to the user's computer or the user can program a remote FPAA system via email to realize mixed-signal systems in the classroom.

devices in silicon. For students to take measurements they would need access to a platform that can realize their large designs and allow them to continually perfect their design choices. If this platform is portable, there will hopefully be less monetary expenses for instruments and personnel to maintain a designated equipment room. The ability to have students see experimental results (i.e theory in action) before leaving any learning environment is valuable and desired.

Figure 16(b) depicts our approach of having a set of RASP 3.0a FPAA SoCs and RASP Tools (detailed in Chapter 2) to incorporate laboratory projects that can be completed inside or outside of the classroom. The FPAA SoC boards are capable of realizing analog, digital, and mixed-signal systems in the classroom and beyond. RASP Tools is a design automation platform for high-level simulation as well as compilation to physical hardware. The tools support retrieval of data through a local board connected to the user's computer or via email that communicates with a remote setup that is described in Section 3.4. Ideally students are only responsible for bringing a computer to load the tool suite and attach the FPAA via USB that is provided.

Our strategy combines and improves some aspects of other researchers' implementations to increase student learning and to deepen their understanding. We were motivated to build our tools and hardware for education when we discovered a change towards student-centered learning facilitating positive outcomes; it was shown that portable electronics increased students' interest in electrical engineering, improved students' confidence in their ability to design circuits and systems, and enhanced students' development of deeper understanding as they investigated core theories and principles through hands-on activities [4]. As stated, FPAAs are configurable devices that can implement different circuit and system topologies to realize varying applications. These boards are an improvement over other design resource kits and prefabricated boards developed to facilitate specific laboratory exercises in class [13, 61].

3.2 FPAAs in Education

FPAAs have been incorporated in an educational setting, ECE 6435, for several years to educate engineers to design for system applications [17, 36, 38, 39, 79, 81]. The evolution of this course has experienced a time prior to FPAAs when pre-fabricated chips were used for each assignment and bulky measuring equipment infrastructures were required [36]. FPAAs and RASP Tools has permitted the structure of this course to include flipped classroom approaches as students use their own laptops and the provided hardware and software for activities. FPAAs alleviate the need for time consuming work like creating multiple PCBs for in-class assignments, homework, demonstrations, or projects. FPAAs, with nearly 0.5 million parameters, enable a wide range of configurable and programmable SoC embedded system computing options. Remote testing is another feature of this toolset that provides an avenue for students to obtain data outside the classroom [73]. The maturation of the FPAA architecture has caused the coverage of core analog concepts and advanced topics to expand and give more freedom to the class to experiment with their own ideas.

3.2.1 Previous Course Development Using FPAAs

The FPAA SoC device has eliminated the requirement of significant bench infrastructures for students to obtain data. It has also enabled the present innovative teaching approach used in recent years. Previously, scheduling concerns for testing and measuring custom ICs were troublesome. The curricular development of ECE 6435 has progressed over the following semesters: Spring 2012, 2014, 2015, and 2016. In Spring 2012, we utilized earlier FPAA devices and an initial toolset developed in MATLAB/Simulink [38]. Within the Spring 2012 class, we also started incorporating design and hardware verification because the FPAA platform was flexible. We then began using the FPAA SoC architectures and tools as we solved some of the prior tool and hardware issues.

Table 1: A comparison of the types of projects and the allocated duration to complete each one as the course curriculum changed over time. FPAA improvements and tools impacted the material discussed, particularly enabling higher signal processing concepts in a given semester. From Spring 2012 and onward, all classes had final design projects take place during the last 4-5 weeks of the course.

Fall 2006		Spring 2012		Spring 2016	
Typical Topic Progression	Weeks	Typical Topic Progression	Weeks	Typical Topic Progression	Weeks
MOSFETs + FPAAs Intro	2	Transistors	3	Introduction to FPAA	1
MOSFETs: Gain + Amplifiers	2	Amplifier		2 MOSFET circuits, OTAs	1
FG Intro, Program, switches	2	Circuits	2	OTA Circuits / Filters	1
Program Current Sources	2	(Integrate & Fire, WTA)		Developing Xcos model	2
Differential Amps + OTAs	2	Transistor	3	(macromodeling)	
Integrate + Fire Neurons	2	Channel Neurons		Transistor Channel Neurons	2
Second-Order + Cochleas	2	VMM + Basic	2	Analog Classification	1
Final Exam, No final project		Classifiers		(VMM +WTA)	
				Dendrite Model + Compute	2
Total	14	Total	10	Total	10

Spring 2015 saw the full integration of our current tools. The lessons learned made the Spring 2016 class operate smoothly with this new infrastructure. The students used these tools and hardware right from the beginning of class and took measurements during the second class meeting [See Appendix C]. These classes benefited from an inverted classroom format, where multiple taped lectures extended time for in-class experimental measurements. Students, typically in groups of two, focused on designing and experimentally verifying their project designs. We had the students work in small groups to provide them accountability, motivation, and responsibility when completing a laboratory assignment. Starting in Spring 2015, we introduced using a remote FPAA SoC test setup [73] that is consistent with the discussion in Section 3.4.

Table 1 illustrates a comparison between the course topics covered for three different years. An evolution of our technology transformed the material taught from 2006 to 2016. It is clear the 2006 course, the first to heavily use a FPAA device, went through a fraction of the core projects that the 2016 class completed. Specifically, the 2006 class barely went through the first 3-4 weeks of the 2016 class assignments and they did not have the opportunity to participate in a final design project (instead,

they had a final exam). The 2016 class benefitted from multiple concepts (*i.e.*, FG devices, programming, and low-level infrastructure) being abstracted to a high-level tool suite. In recent years (2014-2016) the emphasis has shifted from a few multi-week projects to weekly projects; this strategy helps build and maintain scaffolding. In the 2016 class, the two-week labs correspond to the time period when other classes typically scheduled exams. Figure 17 depicts some of the topics and circuits of the experiments for the 2016 class.

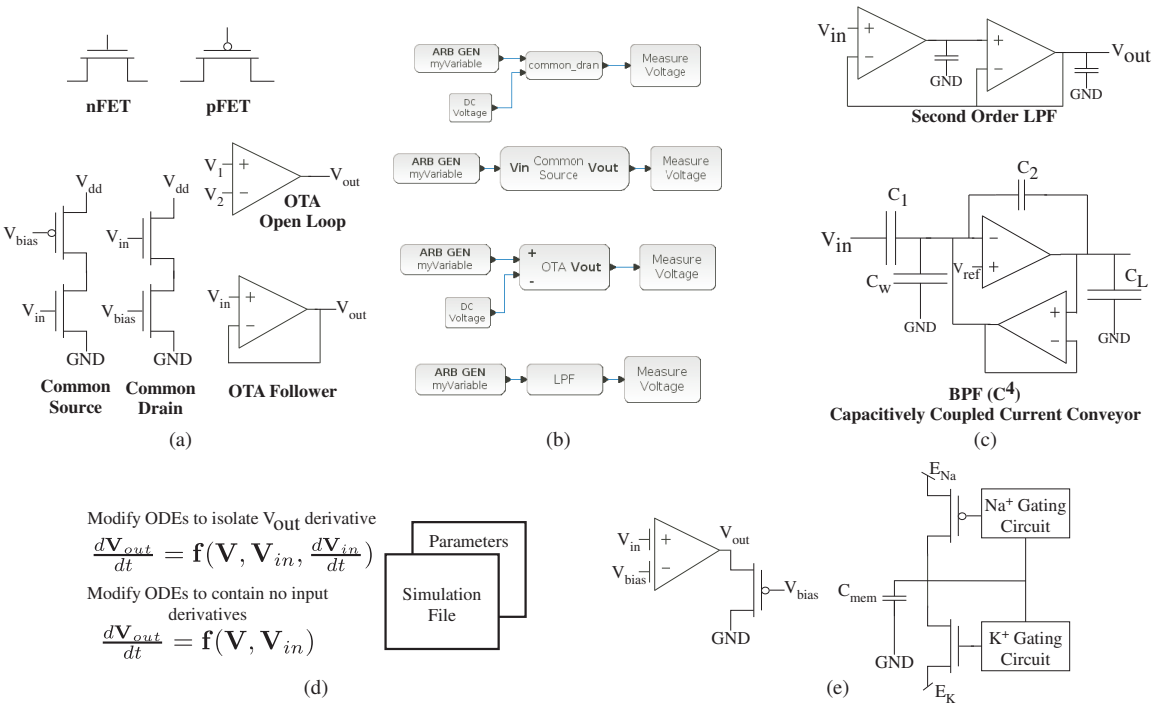


Figure 17: (a) Lab 1: Circuit configuration of pFET and nFET transistors, common-source and common-drain amplifiers, OTA open-loop, and OTA follower (LPF). (b) RASP Tools implementation of circuits used for Lab 1. (c) Lab 2: Circuit configuration of a second order low pass filter (LPF) and C^4 band pass filter (BPF). (d) Lab 3: Determining the macromodels of circuits for the simulation function of blocks by deriving ODEs in the proper form. (e) Lab 4: The transistor channel model and Hodgkin–Huxley neuron circuits adapted for FPAA.

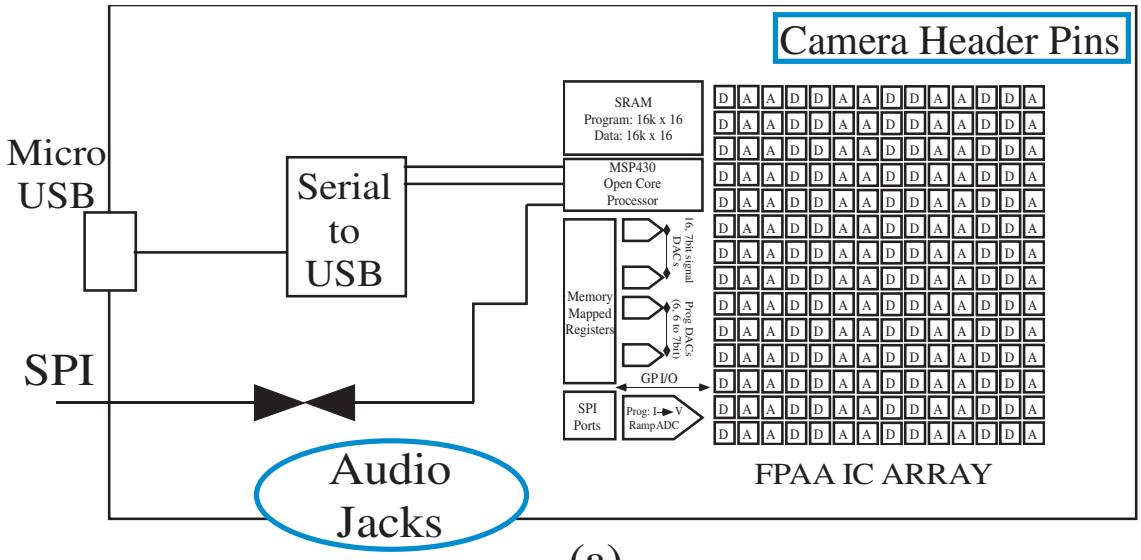
3.2.2 Preparation for Class

Countless hours went into building up the hardware and software to its current form for ECE 6435. Ideas from previous board designs influenced the final design of the

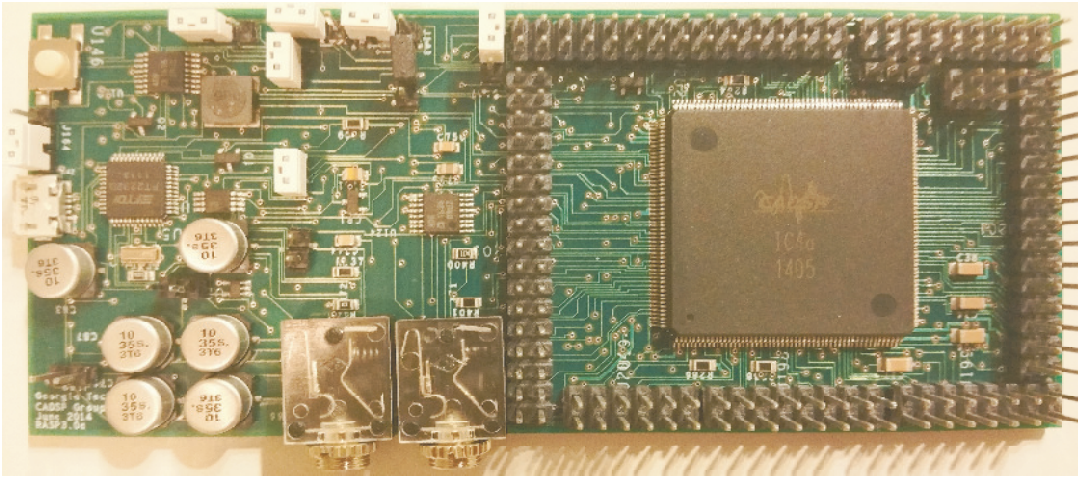
RASP 3.0a board for educational purposes [See Appendix D]. Multiple PCB iterations were completed before sensors were included. These sensors would allow us to expand the variety of projects assigned and the types of applications the students could build when desired. Two sensors were incorporated, namely audio ports and a camera. The audio jacks enable audio signal processing endeavors. An input signal from a microphone can be delivered into the FPAA array for manipulation and the waveform can be played on a speaker to hear the resulting differences. The camera module would be useful for an embedded systems project involving unmanned aerial vehicles (UAVs) for tracking and navigation. Similarly, the camera enables the FPAA to do image processing more directly.

Earlier FPAA SoC PCBs were two detachable pieces, the IC and the control component, joined together by board to board connectors. The layout of the RASP 3.0a board was transformed into a single board [See Appendix E]. Figure 18 depicts this particular FPAA SoC board. With a single smaller board, we did not have to worry about connections between the μP and FPAA array being interrupted due to the connecting header breaking between the two individual boards. The primary off-chip infrastructure is μP IC controlled high-voltage power handling (12V and 6V charge pump ICs); these components were left off chip to minimize the IC design risk, but require additional board level infrastructure. A USB to serial converter IC was chosen to interface with the μP. The USB device is connected through serial interfaces on the board; in our case, we have the potential of a simple serial (8n1) debug interface. The PC board files are openly distributed at http://users.ece.gatech.edu/phasler/PCboards/index.html

Figure 19 shows a detailed pin diagram of the FPAA SoC board used in the classroom; the user chosen FPAA system technology file specifies the internal configurability of an IC to the I/O pins shown. Students that import the provided VM only need this hardware to have a self-contained programmable and configurable platform for



(a)



(b)

Figure 18: Overview of FPAA hardware. (a) PCB block diagram. The board consists of a micro USB connection (used for communication, data, and power), an embedded μP in the FPAA fabric, and other components used for educational tests as well as controlling the FPAA board. (b) Picture of the FPAA Board (2.5"×5"). We use a 208 pin QFP package; many pin headers are connected to FPAA I/Os, DAC outputs, ADC inputs, control pins, power/ground; and stereo audio jacks (on the lower left).

analog and mixed-signal design. RASP Tools is able to communicate with the board for programming and testing through the USB port. This open source reference board design and VM encourages community (students, researchers, and interested users) around these tools and allows this collective group to further improve this toolset.

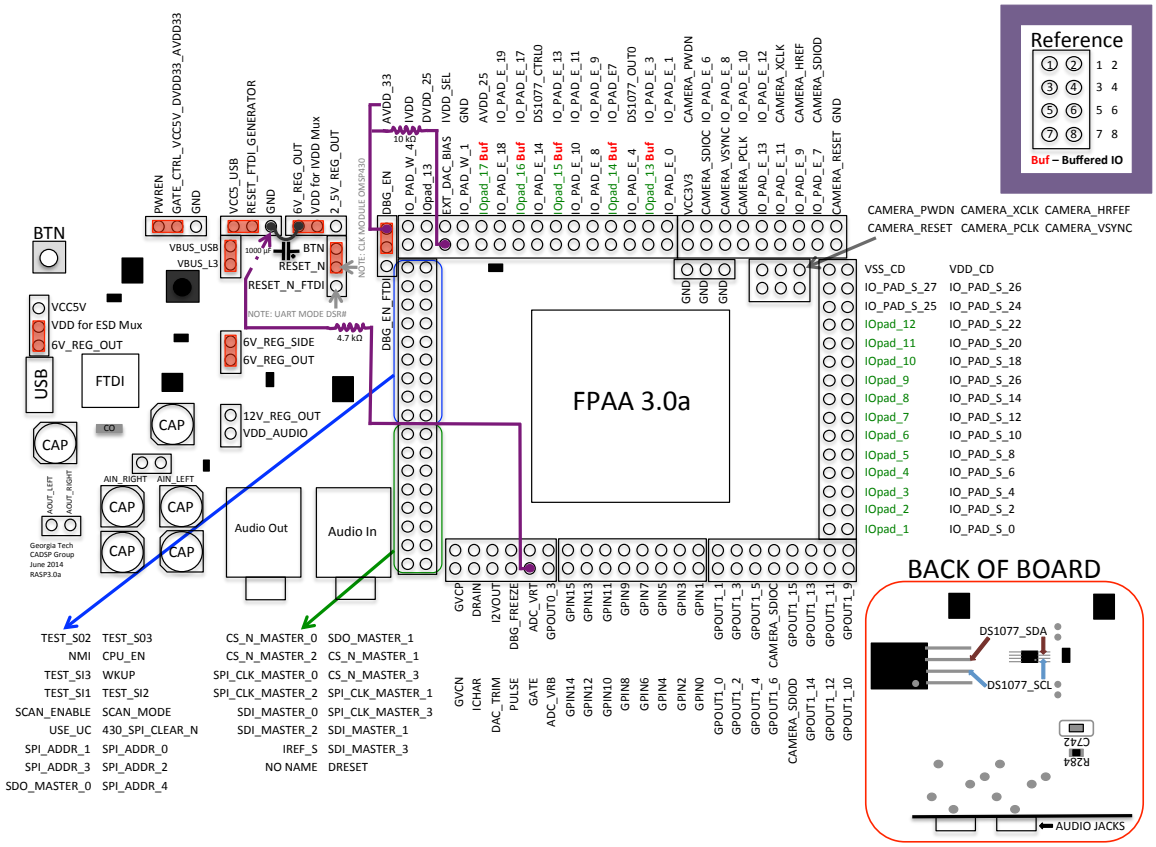


Figure 19: RASP 3.0a Header Pin Diagram for students and users alike to see the layout of the PCB and the location of I/O pins to take measurements. This FPAA SoC board has a combined control and IC board, which is different from the RASP 3.0 board. The numbers printed next to the I/O header pins that are on the periphery of the IC are simply entered by students into their Xcos design.

The RASP 3.0 and RASP 3.0a chips have a tiling structure and size that is dissimilar. Their buffered I/Os, DACs, ADCs, and memory-mapped I/Os are also placed in different locations. Thus after designing a board, we also needed to create new technology files (*i.e.*, python and xml files) that described the architecture of the IC and modify RASP Tools to allow users to select it on the main interface GUI; the distinct mapping in the technology file enables generation of an accurate switch list. The low-level tool *vpr2swcs* that was developed to recognize analog circuits in a target BLIF file, to utilize global and local routing resources for computation, and to accurately map a design for a given FPAA chip to create a switch list using the user specified technology file was also updated. Since **sci2blif** invokes *vpr2swcs*, we

had to make a provision within the tool framework for the new technology files to be recognized correctly. As an aside, the VPR tool that *vpr2swcs* uses for global placement and route also enabled the development of a user placement algorithm (*i.e.*, specific placement of a block to any FPAA array location [x,y]) [See Appendix B Algorithm 3]. Thus, a user can restrict individual blocks in their Xcos design to a particular "row and column" within the FPAA IC array.

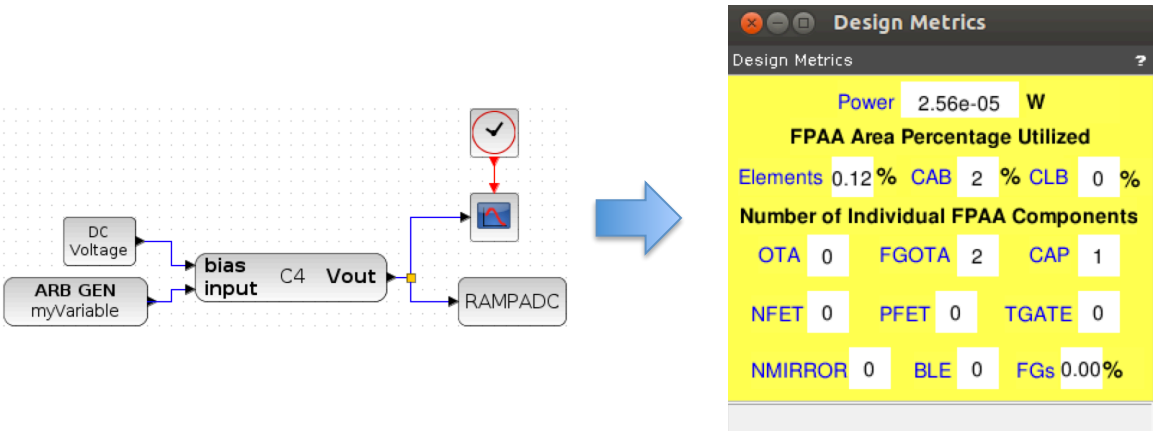


Figure 20: An example using a C^4 to illustrate the design metrics GUI. It is an interface that displays useful metrics to the user to make informed changes to their design if desired and have comparison points across designs to choose the best option. The interface gives the user power, area percentage, number of CAB and CLB components, and FG usage percentage values.

In addition to hardware preparation, software improvements were made to assist users. Figure 20 depicts an added feature to RASP Tools; a GUI that informs students of their design choices. The interface gives the user the following values: power in watts (most designs will be in μW range), area percentage (considering total CAB and CLB components of FPAA fabric and total number of CABs and CLBs), number of individual CAB and CLB components used in design implementation (OTAs, FG OTAs, Capacitors, nFETs, pFETs, T-Gates, Nmirrors, and BLEs), and FG usage percentage. With the provided information they can optimize for lower power, smaller area, and component use as well as compare circuit topology options. The underlying code of the GUI takes into account which board is being used in order to adjust the percentages shown because some parameters for calculation will be different.

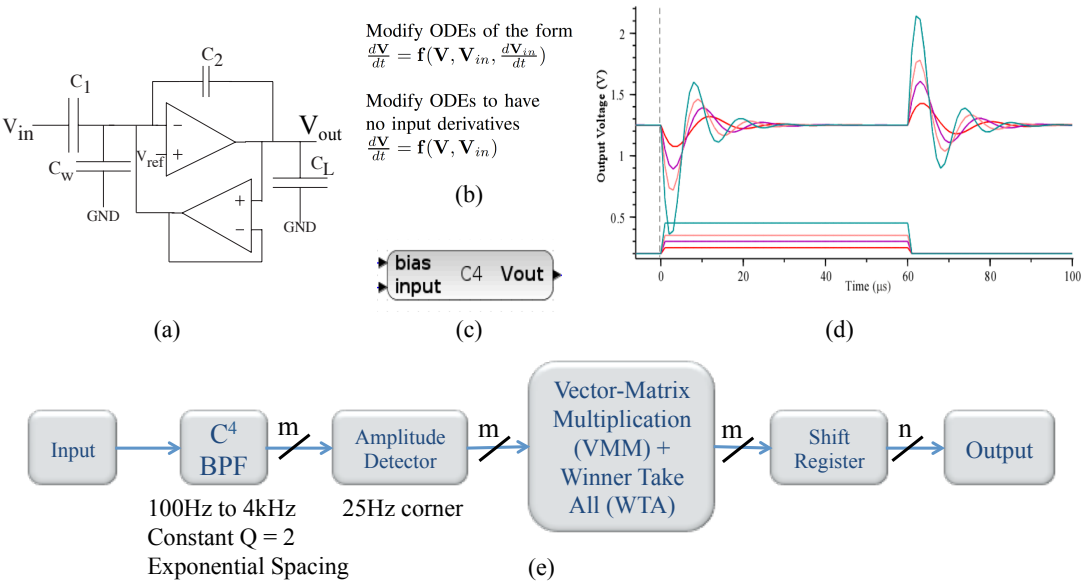


Figure 21: Analog block reuse for larger systems. After obtaining measured experimental data, a corresponding block should be created that simulates and compiles to hardware. (a) Circuit diagram for a C^4 band pass filter. (b) Manipulating ODEs to have the final Xcos formulation. (c) Xcos block that is linked to the ODEs and compilation code. (d) Simulation of a step response for the C^4 band pass filter. (e) The new C^4 block used in a larger design, which can again be encapsulated in a block for reuse.

3.3 Circuit Blocks Reuse

Analog IP reuse is a concept rarely at the forefront of discussion when talking about analog circuit and system design. This however is commonplace for digital designs. Our software tools enable both. The user can reuse an existing analog IP block to build newer blocks that can be abstracted as standalone blocks to place in their own palette. We illustrate this concept in Fig. 21 with the example of a C^4 band pass filter. Once a user gets experimental data, they can proceed to make a corresponding block (interfacing and computational function files) that simulates and compiles to FPAA hardware. The new block is now available to be used in a larger design, which can again be encapsulated in a block for reuse. These user generated blocks can be shared among users, which supports collaboration in the class and more importantly a vibrant analog and mixed-signal design community.

3.4 Remote System

There are instances when individuals of a group may want to take data, but their group-mate has in their possession the device they need. We have seen a wide range of remote test systems, which have spent considerable time developing their hand-tailored configurable systems [14, 15, 20, 45, 56, 76], to address this issue. In our case, we wanted to provide a means to let students obtain data remotely (*i.e.*, without a physical board connected to their computer) through RASP Tools. This system will offer students flexibility in completing labs during times that best fit their schedule.

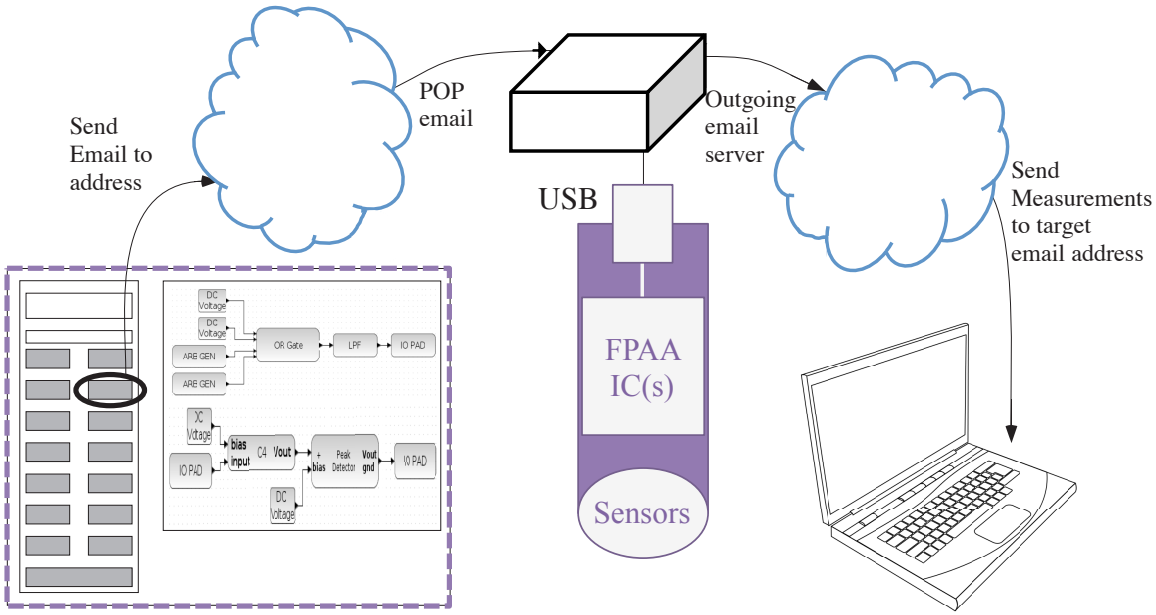


Figure 22: Detailed flow for the remote test system implementation. The design toolset in Scilab/Xcos allows the user to send an email to program and measure a remote FPAA IC. When that option is chosen, the resulting file is sent by email into the cloud. The email is retrieved from the server using the Post Office Protocol (POP), the programming files are extracted and executed, the data measurement is performed on the device, and the results are sent back by email to the original sender. The user can directly use the results in Scilab or any other data analysis program to observe their data as well as complete their analysis.

Figure 22 shows the framework for our remote system approach. We wanted our structure to be as easy to use on both the user and remote server side (*i.e.*, requiring minimal user maintenance, using an integrated tool platform, and having few location

constraints as possible). We utilize standard email servers to enable a relatively stable remote platform capable with nearly zero administrative overhead. The remote server periodically checks for email on the server, downloads the message using Post Office Protocol (POP) from the server, checks its control syntax, and has the object code ready for programming the IC. The student on the other end, would simply need to enter his or her email address in the main interface GUI of RASP Tools to send their circuit or system to be tested on the remote device instead of their local board. The *Send Email* button when pressed emails the compiled design files to a location where the remote system can access it and send a return email with experimental data after the design is programmed and executed on the remote device. The benefits of having this system in place are the accessibility, equipment cost reduction, scalability, automated measurements, and collaborative opportunities with other institutions.

3.5 Summary of FPAAs in An Analog Course

The use of FPAAs in a graduate level analog course has seen many changes over time. The first FPAAs highly relied on equipment workstations that posed scheduling issues for students. The current FPAA SoCs are self-contained as they have internal measurement modules; these FPAA SoCs are operated through USB connection to a computer with RASP Tools. Students also have the option to connect other instruments to their boards. RASP Tools was improved to handle the RASP 3.0a IC for class in addition to the RASP 3.0 IC that the tools initially accommodated. As a result, students can make and store blocks in their own palette and share their IP with classmates to design more advanced systems. Similarly, students have control over the placement of their blocks within the FPAA array and can decide whether to experimentally test their design locally or remotely. In the next chapter, we discuss the assessment of using FPAA SoCs in the graduate analog course ECE 6435.

CHAPTER IV

FPAA AND RASP TOOLS IMMERSION ASSESSMENT

This chapter presents the assessment results of using FPAAs and its associated design automation software, RASP Tools, in an analog graduate level course to integrate hands-on activities for learning. We describe our teaching methodology as well as experiments involving the FPAA SoC. We are evaluating the student satisfaction of using RASP Tools and FPAA SoCs for analog design and our blended approach to convey this material. Metrics considered are students' perception of hardware and software capabilities, self-efficacy in the core areas, and their assessment of the course methodology.

Our approach to assessing the impact of students' learning and acceptance of the technology is through pre- and post-surveys and classroom discussions throughout the semester. This class uses a blended approach to facilitate learning through pre-recorded mini-lecture videos, portable laboratories, traditional lectures, classroom discussions, and in-class exercises as illustrated in Fig. 23. The technologies used during this course are FPAA SoCs [32] and its design synthesis software RASP Tools [18], which includes programming capability [44]. The assessment methodology employed is shown in Fig. 24. The size of the class during pre-survey was 15 students and during post-survey was 8 students.

4.1 Teaching Methodology

Figure 24 illustrates how we assessed the students' progress towards course goals throughout the semester via student videos, class conversations, circuit demonstrations, and in-class exercises. The measures presented in Fig. 24 were also used to evaluate our course approach effectiveness. Figure 25 shows core areas like analog

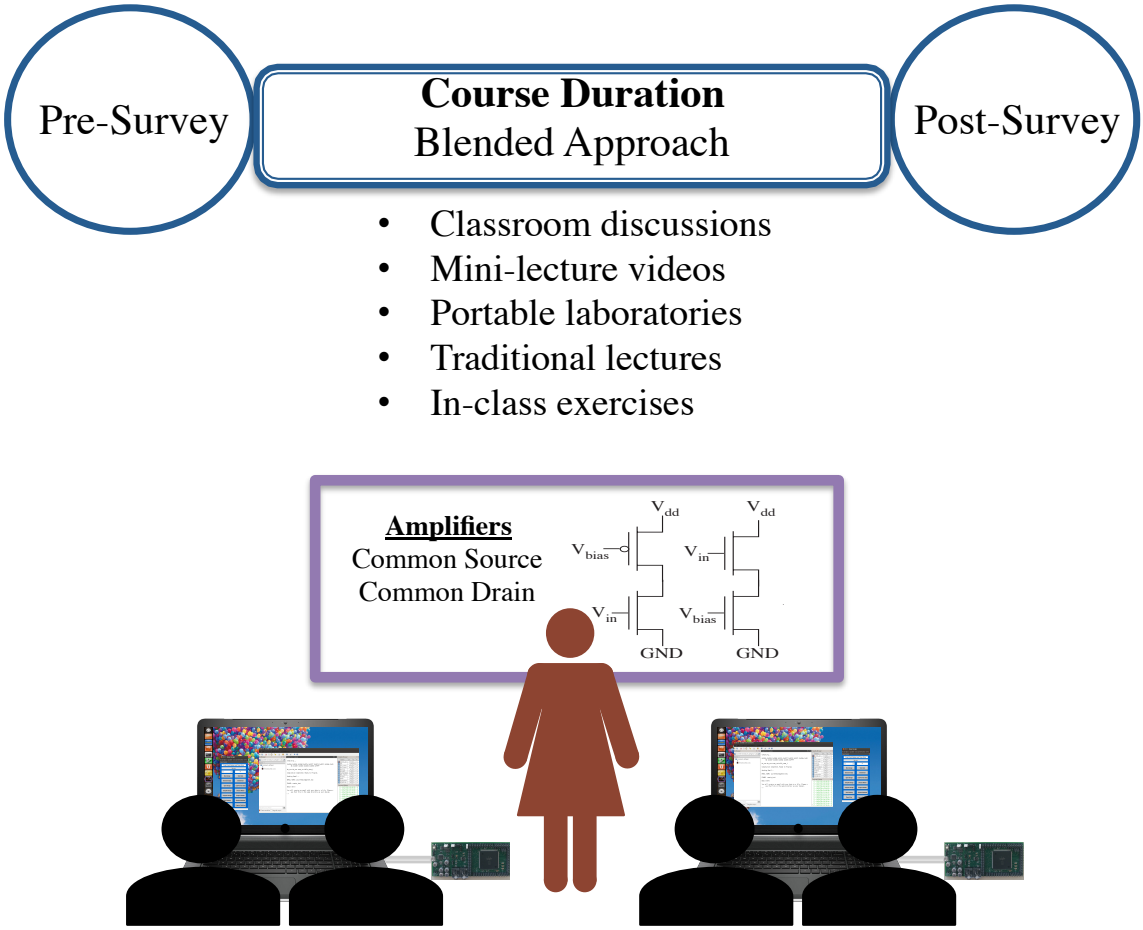


Figure 23: Overview of approach to assess using technology in an analog course. Using hardware and software namely, FPAAs and RASP Tools, we created a blended course to facilitate learning through hands-on labs. This created an environment that promoted interactive class discussions. Pre- and post-surveys were given on the first and last day of class, respectively to evaluate the course.

circuit and system design, model-based and modular design, signal processing, and neuroscience that were covered in this course with intention. We paired in-class lectures, discussions, short videos, and experiments to create a blended course. The course objectives were defined as:

(a) Students will be able to design neuromorphic analog circuits and systems.

(b) Students will be able to analyze neuromorphic analog circuit and system data from FPAA.

(c) Students will be able to recognize, relate, and verbalize analog concepts to

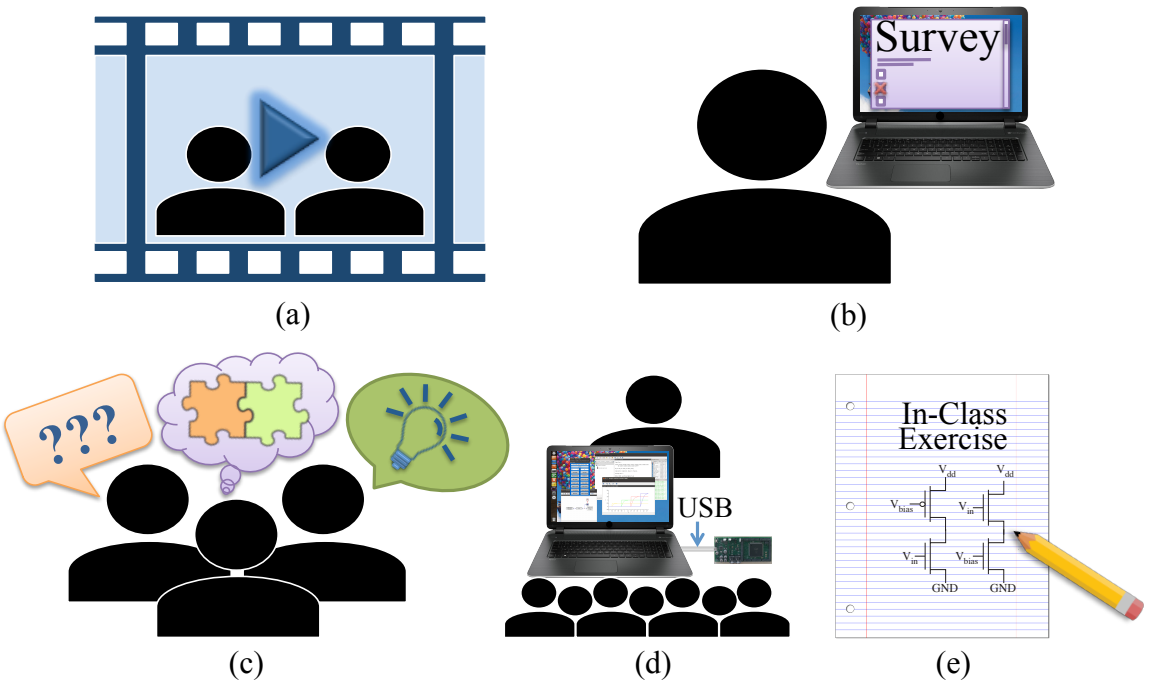


Figure 24: Assessment techniques used to gauge students' comprehension of course material. (a) Five-minute video of laboratory project results, observations, analysis, and rationale. (b) Student self-reporting through pre- and post-surveys about their learning and interaction with FPAAs and RASP Tools. (c) Classroom discussions concerning analog and neuromorphic circuits driven by instructors and students. (d) Student demonstrations of working projects using two modes of data acquisition. (e) Non-evaluative quizzes of circuit concepts.

phenomenon observed in data.

(d) Students will be able to use intuition to create novel implementations of neuromorphic analog circuits and systems in FPAA.

(e) Students will be able to model analog circuits that correspond to silicon data from FPAA.

The lectures were purposely interactive to encourage dialogue and create an environment of community. The discourse was sparked by students ideas, comments, and questions as well as instructor queries. Five to ten minutes lectures were made available to students at the beginning of the semester to allow students to preview future material and prepare for topics covered before coming to class. We promoted

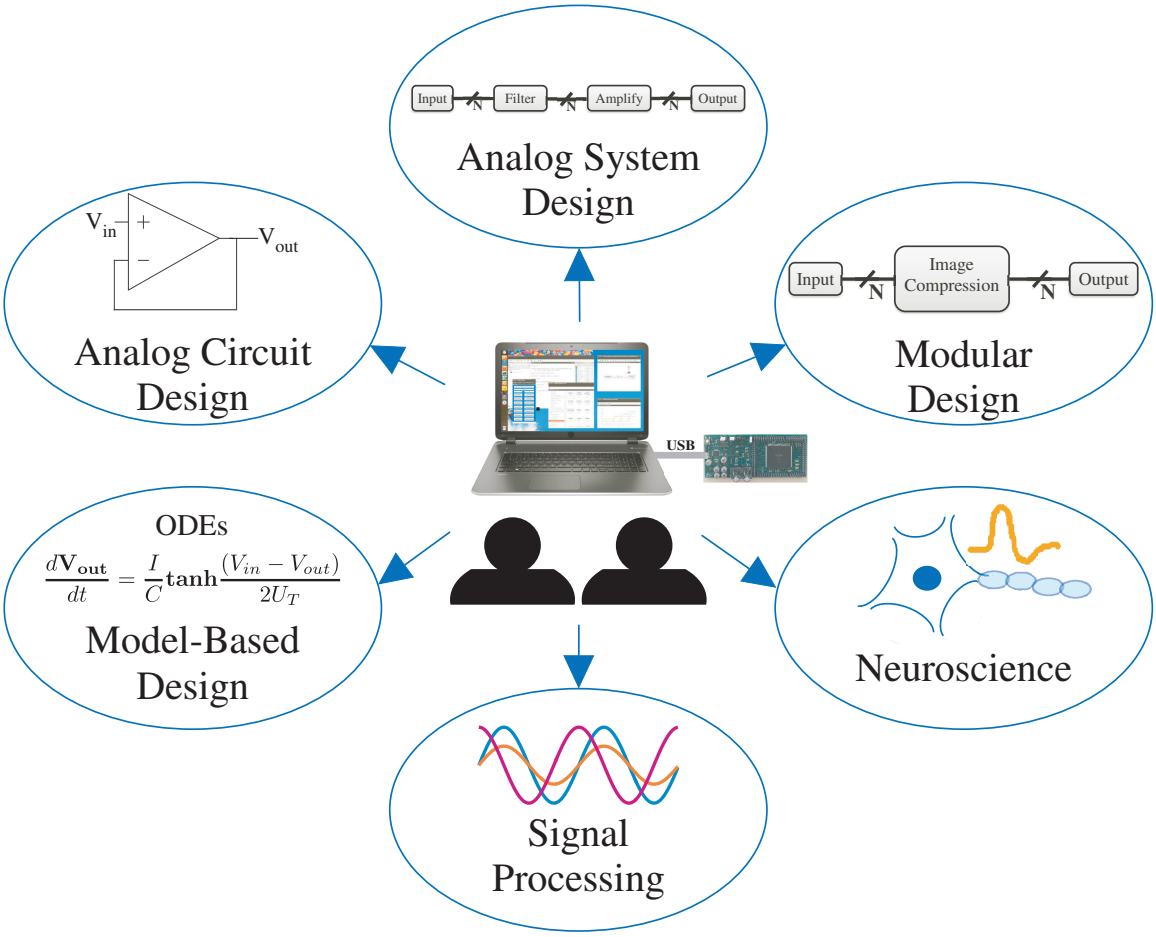


Figure 25: FPAAs and RASP Tools are designed to allow students to focus on circuit operation using a single platform that enables them to revisit previous circuit designs while learning new material. Using this configurable hardware, educators have the ability to vary course content, build on foundational concepts, and explore system designs. In this course we were able to delve into multiple areas like analog circuit and system design, model-based and modular design, signal processing, and neuroscience.

"learning by doing" in our laboratory experiments which are discussed further in Section 4.2. We had students record project videos that were released to the class for review by next meeting time. In this course we gave students FPAAs and virtual machines with preloaded software, RASP Tools, to install on their computer. RASP Tools is the FPAA's design synthesis tool that enables simulation and experimental measurement.

Acknowledging the classroom is a place where student perception of material and the desire to pursue more knowledge can be influenced, we developed laboratories for

this course to increase the students' familiarity of analog design concepts and experimental practice. Each group was assigned a FPAA board to use for the semester. Since each FPAA board has unique characteristics (transistor mismatch), we expected that students would not obtain identical values and graphs with the same parameters. Each group was expected to record a short five to seven minute video of their process, results, and analysis. The videos were uploaded for the groups to view each others work a day before class. This visual documentation of their procedure to solve problems and analyze data will be useful in the future for new cohorts. During the next class period we were able to discuss common issues, delve into other interesting technical topics, and even explore ideas outside the scope of the course. Students transitioned from merely stating their observations to analyzing their data more in-depth to account for non-idealities in their video for each lab. By the end of the semester, a student that did not have an analog background decided to expand their final project progress into the summer.

In a traditional classroom, students are lectured to for the duration of the class without formative assessment exposure or participating in active learning exercises. A course where students do not practice their comprehension of the material until they take a form of summative assessment, which may be the midterm, is not beneficial for students. By not addressing the topics that the class misinterprets, especially when material builds upon previous concepts, many students may become overwhelmed and discouraged from progressing further in their major. Attributes like these of a traditional classroom have been challenged for years [10, 78]. Alternative methodologies have been suggested to improve the learning experience of students so that they internalize more information through practice and hopefully real-world exposure [1, 9, 24, 27, 28, 40–43, 51, 55, 58, 63, 74, 77, 83]. Giving students a means to become aware of their conceptual acumen is important. Lectures alone do not provide students valuable feedback, which would allow each individual to assess

their strengths and weaknesses. Consistent clarification of misconceptions through recitation sessions, group discussions, and in-class activities are viable options. Thus, educators have been introducing methods like in-class activities for individuals and groups such as problem solving, games, discussions, and other technology to supplement lectures [5, 13, 15, 45, 47, 61].

4.2 Laboratory Projects

Analog courses with laboratories typically have students obtain data for each assignment with different pre-fabricated boards or standalone ICs, wires, and solderless breadboards that can become convoluted. These laboratory activities require disassembling the previous project to create space and avoid confusion. In this analog course, we focus on giving the class hands-on project experience instead of only circuit simulation based problem sets. Students are exposed to working experimentally with fabricated hardware while designing and analyzing circuits. In this environment, students are able to observe and account for non-idealities that would not have to be considered if only solving equations and running a series of simulations. Over the course of the semester we had a total of six projects, where we separated the class into groups of two.

Lab 1: Transistors and Basic Amplifiers

In this lab, we had students take measurements with the remote system to verify their setup was installed and working properly [See Appendix C]. They showed a plot that compared remote and in-class measurement results. Afterwards, they investigated and regressed nFET and pFET transistor current measurements to determine key parameters such as κ, U_T, and σ. These characteristics aid in combining two of these transistors to form amplifiers, namely common-drain (source-follower) and common-source. The students were able to observe and determine the gain of these amplifiers and mathematically prove their findings. Lastly, the class worked

with an operational transconductance amplifier (OTA) where they used it in both its open-loop and follower topology, Fig. 17(a) and Fig. 17(b).

Lab 2: Low and Band Pass Filters

The next project focused on second-order section behavior, as shown in Fig. 17(c). The class analyzed a second-order LPF that is composed of two OTAs. The students were able to view the increased linearity afforded. The FPAA board is equipped with audio ports, which enabled students to compile a bank of band-pass filter (BPF) blocks to selectively attenuate frequency bands from an audio input and hear the modified waveform.

Lab 3: Macromodeling Circuits

Macromodeling is the process of building blocks and their simulation files. These blocks represent circuits previously made or new ones, where students can confirm their understanding of non-idealities observed from compiled circuits working in silicon. The ODE models they create realize the behavior seen in such measurements, Fig. 17(d). By encouraging reuse of these blocks, which is a concept rarely at the forefront of discussion when talking about analog circuit and system design, students can share their ideas with each other to build interesting systems. Thus, promoting a community of designers to learn from another. Students successfully demonstrated analyzing a neuromorphic analog circuit by creating their own simulation mathematical model for a dendrite and verified it with hardware data.

Lab 4: Neuron

This project focuses on understanding and experimentally measuring the transistor channel model approach for handling passive and active channels. With the experience from the previous lab, students built more blocks concerned with building a Hodgkin–Huxley (HH) neuron. The HH neuron model takes into account the excitatory and inhibitory ion channels, sodium and potassium, that are necessary for an action potential to occur, Fig. 17(e).

Lab 5: VMM and Classifiers

In this project, the students focused on one type of classifier structure, the VMM + WTA that elegantly compiles into our FPAA structure. The XOR function, which is a two layer neural network equivalent, was implemented [32].

Lab 6: Dendrites and Diffusors

The focus of this project was to get a working dendritic line built from a diffusor circuit approach. After biasing approximated changes in a dendrite cable diameter, students combined multiple dendrites together to illustrate a dendritic-modeled neuron classifier.

4.3 Assessment of Pedagogy

Many features of RASP Tools were utilized by the students while completing their lab assignments. The class was able to manipulate existing design examples, view simulations, compile the design to FPAA hardware, and view experimental results. With this experience, students were able to create their own circuit blocks to create new designs. Their blocks contribute to the palette library of pre-tested block modules that translate to circuits on the FPAA board. Throughout the semester, students received feedback from instructors and peers on assignments to augment their learning. We were interested in determining the effectiveness of hardware and tools, the course environment, and the students' confidence of their mastery of course material and associated skills. Therefore, we planned to give two surveys that encompassed inquiries to assess the impact of our approach.

4.3.1 Previous Observations

We noticed former students preferred using the remote system to take project measurements throughout the semester. This occurrence was a surprise at the time because the class had access to FPAA boards and Diligent's Analog Discovery [22]

Table 2
MEAN RESPONSE: HOW DO YOU CLASSIFY YOUR SKILLS IN WORKING WITH...?

Areas of Expertise	Before	After
CAD Tools	2.60	3.00
FPAAs	1.33	2.50
Scilab	1.73	2.50
Xcos	1.27	2.38
Embedded Systems	2.20	2.75
Hardware Debugging	2.87	3.00

Note: NOVICE (1), ADVANCED BEGINNER (2), COMPETENT (3), PROFICIENT (4), EXPERT (5)

data acquisition boards that were either supplied or purchased individually. A considerable improvement was made in the software to include more documentation, measurement support, block modules, and utility features before this semester began. This semester we chose to not introduce the class to the Analog Discovery system to acquire experimental values. Instead, we had the class use the multiple FPAA internal measurements block modules that use on-chip DACs, on-chip ADCs, and compiled ADCs. We wondered if this switch in measurement systems would influence the students to continue choosing to use the remote system over the local FPAA boards or would there be a shift in preference. Our prediction was that the current cohort inclination would be in favor of the remote system. Our opinion was developed by weighing the flexibility the remote access gave the students to not physically be in the FPAA board storage area while obtaining the same results. We discovered the students preferred using the local boards over the remote system.

4.3.2 Pre-Survey

On the first day of class we had students take an anonymous online survey. We chose to give this survey on the first day because it guaranteed they had not interacted with the particular FPAA hardware or RASP Tools that we are assessing. Their initial evaluation of themselves, hardware and software, course structure, and personal

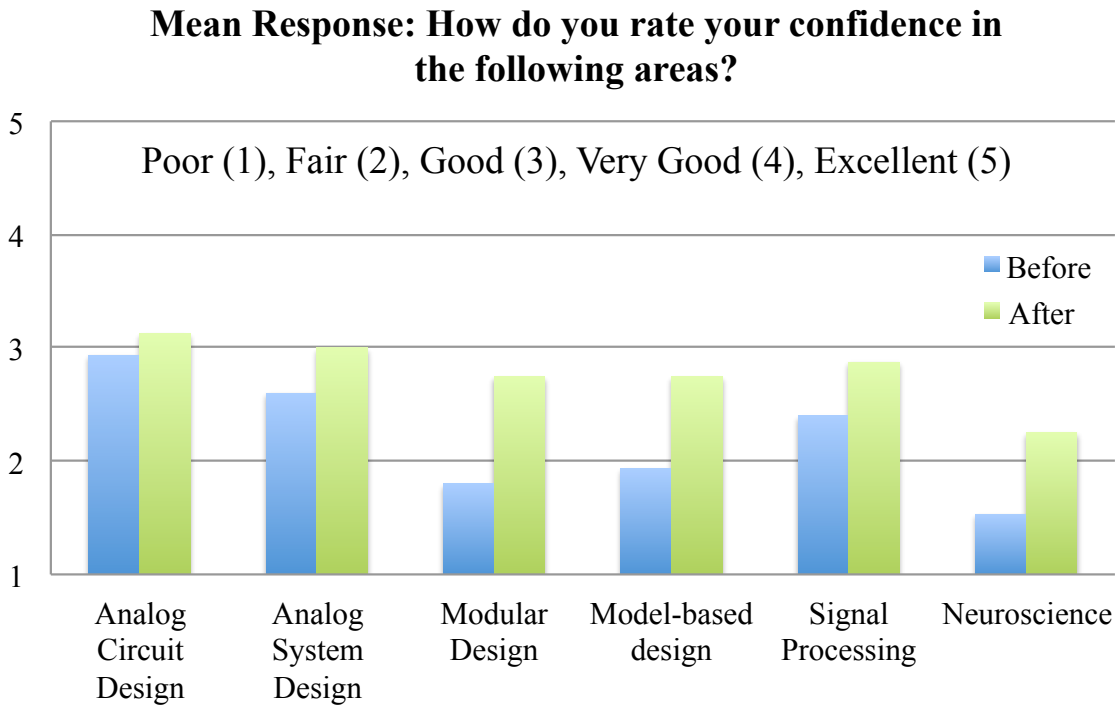


Figure 26: Students were asked to rate their confidence in the core areas of the course. The methodologies aimed to build up student expertise in analog circuit and system design, modular and model-based design, signal processing, and neuroscience.

preferences were polled. Our main reason for not collecting identifying information was to receive candid responses of these topics. Our belief was that the students would be assured their opinions would not be factored in or have an influence on their grade for the course. This survey has allowed us to gauge the previous knowledge, skills, and thoughts to be contrasted to a follow-up survey given at the end of the semester.

This course uses analog circuits to mimic the building blocks for biological information manipulation and processing (*e.g.*, brain and ears). Figure 26 shows that we probed the student's background knowledge of related topics by having them rate their confidence in analog/digital circuit design, analog system design, digital circuit design, digital system design, modular design, model-based design, signal processing, and neuroscience topics. Another question within this realm posed on the survey was familiarity in working with CAD tools, FPAAs, Scilab, Xcos, embedded systems, and hardware debugging, as shown in Table 2.

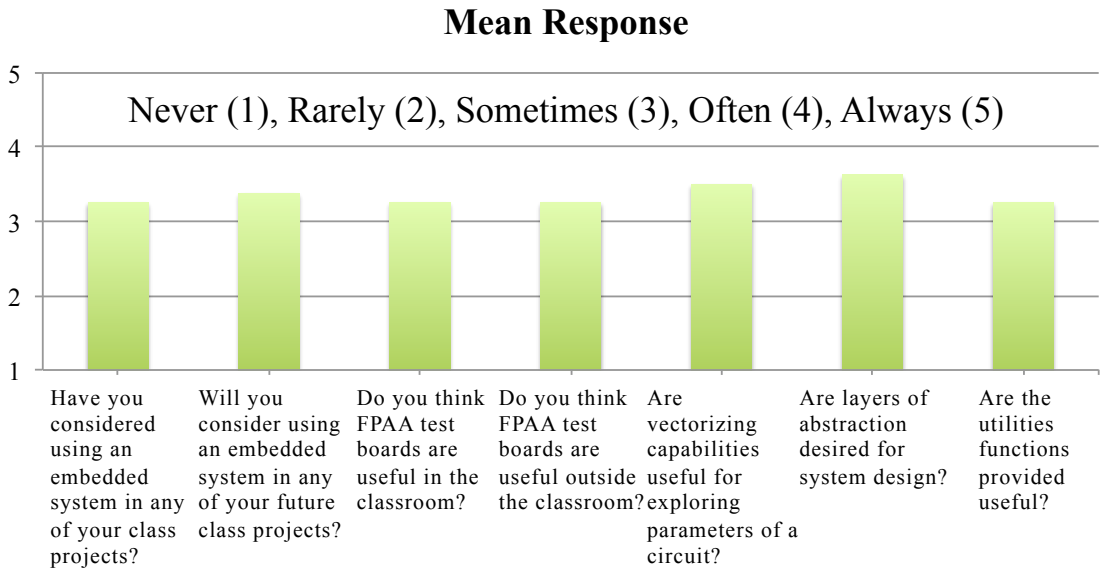


Figure 27: Post-survey results of the students' response to being exposed to FPAAs and RASP Tools.

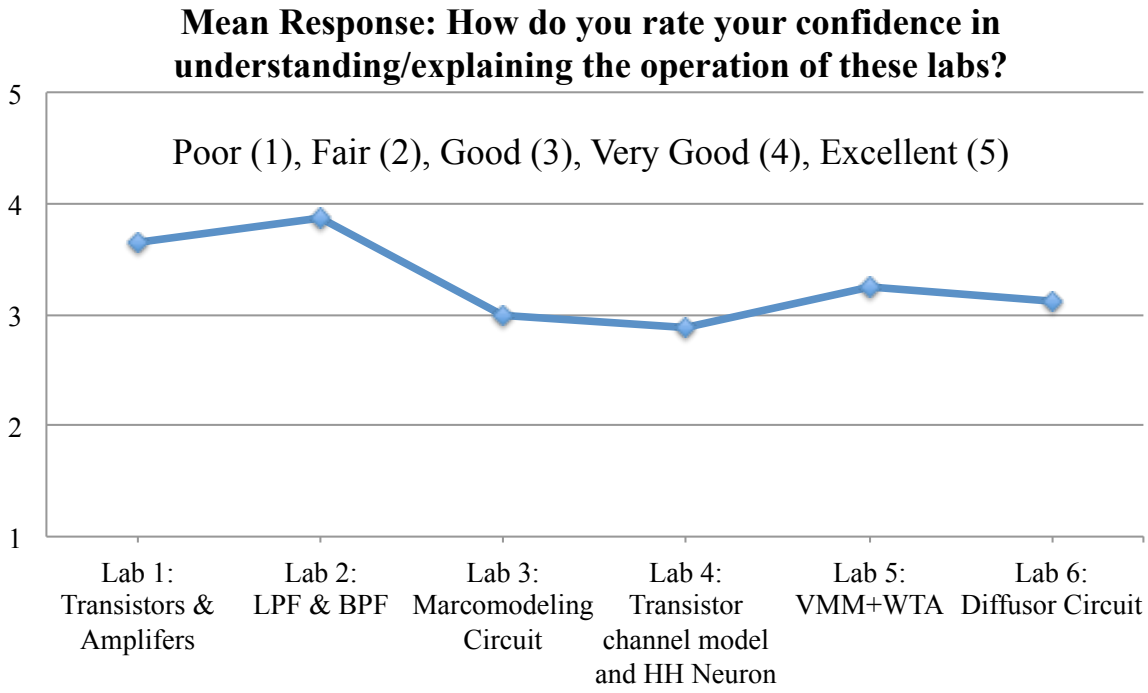


Figure 28: We assessed the class to rate their confidence in each of the six semester laboratory projects.

4.3.3 Post-Survey

Final project oral presentation day marked the last day we would interact with the class and the best time to provide the link to a follow-up survey, similar to the former

Table 3
MEAN RESPONSE: TO WHAT EXTENT DO YOU AGREE OR DISAGREE WITH THESE STATEMENTS...?

Statements	**Rating**
In-class demos improve your understanding of the subjects taught in class.	3.25
Having lab time spent in class improves your understanding of subjects taught.	3.25
Having lab time spent out of class improves your understanding of subjects taught.	3.75
In-class demos help you complete labs.	3.38
Final projects are beneficial.	3.88
Graphical representations of circuits are easy to comprehend.	3.50
You prefer graphical representations of circuits over netlists.	3.75
The projects increased your learning.	3.50

Note: STRONGLY DISAGREE (1), DISAGREE (2), NEITHER AGREE OR DISAGREE (3), AGREE (4), STRONGLY AGREE (5)

pre-survey with additional questions. This survey also did not acquire identifiable information for the reason mentioned earlier. Figures 26, Fig. 27, and Table 3 depict the inclusion of more reflective questions in addition to accounting for the students' final opinion of their skills, the FPAA board and RASP Tools, assignments, and impression of useful features. Our results have provided insight for next steps in improving the tools to achieve our overarching aim and learning in this course.

As illustrated in Fig. 28, we felt it was essential to determine the effect that the lab projects had on the students. Several questions address this objective by allowing each student to evaluate their confidence in understanding and explaining the project material to another person. For example, students begin with transistor curves to discern its operation and extract parameters, then produced macromodel equations to show circuit behavior, and finally used a VMM+WTA to build classification hyperplanes. We then clarified the degree to which they believed that they had mastered procuring experimental measurements and interpreting collected data, which are two

main skills that we desire for the students to strengthen. Other questions evaluated the remote system, measuring apparatuses, project effectiveness, embedded system use, and tool features. We inquired to what extent FPAAs are useful, vectorization is appropriate for parameter coverage, and abstraction is desired for system design.

4.3.4 Comparison and Trends

We asked the students to classify their skills in the following hardware and tools at the beginning and at the end of course: CAD Tools, FPAAs, Scilab, and Xcos. The survey shows that the students had some familiarity with CAD tools before, but did not know about FPAAs or the tool Scilab. Table 2 shows that the students' competence increased in each of these areas by the end of the semester. If we look at Fig. 26, we are able to view the students' rating of their confidence in using hardware and software at the beginning and end of the course. We notice that the students are more comfortable using CAD tools by the end of the course, which can be attributed to the number of labs given. Those who identified as advanced beginner and proficient increased their skills to a higher expertise category. A majority of the students had little confidence in interacting with FPAAs, Scilab, and Xcos at the beginning of class. We saw a transition from novice to advanced beginner and competent consistently across the FPAAs, Scilab, and Xcos graphs. Knowing that the group sizes were small and that members worked closely together could explain this trend of similar improvement. We want to note that by giving the first survey on day one of the class before the end of the add/drop period, the sample size differs from the final survey totaling in at fifteen and eight responses, respectively.

We polled the students to rate their confidence in: analog circuit and system design, modular and model-based design, signal processing, and neuroscience. Students had some familiarity with analog design, but were not proficient in signal processing, modular design, and neuroscience. Figure 26 shows that the students' perception of

their competence increased in each of these areas by the end of the semester, where no student assessed their ability as poor. The course was designed to cover aspects of each of these areas. Figure 26 portrays that everyone in the class felt they understood analog circuit and system design at least fairly; there is no major difference in the breakdown of the knowledge level across the categories. It was good to see that no one labeled their confidence as poor after completing the projects during the semester. Modular design is related to taking existing circuit block modules from the palette library to make a system, while model-based design can be mapped to the simulation models created and performed; the class gauged their confidence to be fair and good, which lends to the idea that if we were to give them an analog system to design they would know how to approach the problem. The signal processing data is related to the BPF lab where they were able to manipulate an incoming audio signal by tuning out particular frequency bands. The neuroscience concepts are related to the bank of filters used to mimic the ear as well as the neuron and dendrite labs that focused on the transfer of information. These labs had an impact because no response was in the poor confidence category. It was interesting to notice signal processing skills improved more than neuroscience. In Table 3 we depict the effectiveness of the teaching techniques implemented in class. We see that students appreciated in-class demos and labs and felt that they contributed to their success in the class. Students enjoyed working with graphical representations of circuits and it aided their learning of the subject material. The students appreciated in-class activities including designated lab time and liked that they were working with graphical representations of circuits and systems over netlists.

Figure 27 depicts the opinions of students after interacting with the tools and hardware. The students considered two features of RASP Tools, abstraction and vectorization, to be desired and useful as well as other key integrated attributes. The FPAA hardware was deemed suitable for use inside and outside of the classroom.

The laboratory projects that we had the class complete ranged from circuit basics to complex classification and neuromorphic circuits. We asked the students to reflect on their process of finishing these labs and to determine how much they learned. Figure 28 details the cohort's perception of their level of understanding and their ability to explain the topics of each lab to another individual. The graph depicts a mostly high student competence as a majority of the responses were in the good and very good categories. We soon learned that no one in the class thought they had a poor grasp of any lab material. We also became aware that some students considered themselves experts or very well-versed of some lab topics. The trend revealed from evaluating this data is that the majority of the students believe they have confidence in their expertise. These results were very encouraging and showed that these labs improved their overall understanding of the course content.

Tool and Assessment Incorporation

Throughout the semester we conversed with students to understand their concerns for the tools and the class. We became more aware that the block library in Xcos should be more extensive, that more examples that vary in complexity would be valued, and that various forms of formative assessment during the course like "Think-Pair-Share" and one minute papers would enhance the community we strive to cultivate. In the case of FPAAs, we can leverage the routing switches or resources because they are not to be considered useless. VMMs are easily created in analog hardware through current summation of transistors in routing. VMMs are useful in various applications including audio signal processing as well as image convolution and classification using bio-inspired circuits. As the size of the applications vary, so will the size of the VMM. Instead of having users choose from preset sizes, we want to improve the tools to build and handle user specified sized VMMs.

4.4 Critique and Considerations

The results of this study are not free of limitations. One of the things we could improve is how we handled the identity of students. Unfortunately, we could not compare students individually before and after, as the students did not remember the initial random number allotted to them. This led us to rely on aggregate data for assessment instead of an apples to apples comparison as planned. Also, anonymity of students for a survey encourages them to be forthright of their feelings, without worry of repercussions. Instead of giving students a random number, we could have let them choose a number combination that they would remember for the pre- and post-surveys. The survey could also be accompanied by a glossary of terms that clearly stated the definitions of language used in questions. This would leave less room for ambiguity. Also, an even number of options would eliminate the mean response trend we saw so that we knew more definitively if they agreed or not. In addition, instead of relying on student self-evaluation, we could use a survey that could gauge their knowledge similar to a concept inventory test. We agree that the questions could be more objective and not be leading to a particular response.

The software and hardware were relatively stable this past Spring 2016, making it ideal for a tool immersion assessment evaluation of the course. We wanted to study the effectiveness of key features like vectorization, tool environment, and circuit abstraction. Due to the small sample size of the post-survey, it was expected that the standard deviations will be large. The post-survey was valuable to ascertain the students' satisfaction with the tools. We also learned that getting survey data via the internet was very useful for data collection and evaluation. Through abstraction of analog circuits we strive to reach a larger audience. The goal is to not scare users with details and to give them an initial high-level understanding of the material, which would encourage them to explore the subject more. We are using a top-down methodology. One might argue, to what degree is it good to remove levels

of abstraction and in what cases can it be harmful? After all, an excellent analog designer is an artist. However, by decreasing the initial potential barrier to learn analog design, we can attract more people to analog design which is considered a niche area. The advantages will be twofold: more creativity in design due to diversity of users as well as making analog design ubiquitous. The goal is to attract new users and retain them through analog education. The tool also provides avenues for users who want to further develop their skills.

4.5 Conclusion: Initial FPAA and RASP Tools Positive Impact

We learned that students have a positive perception of the capabilities of FPAAs and RASP Tools, their self-efficacy in core areas improved, and their assessment of the course methodology was favorable. Our tool facilitated the completion of projects by providing essential features while aspiring to minimize negative experiences. The students' informal comments have been noted and will guide the inclusion of more features and block modules to enhance the software. The moderately affirmative impression the class had of the hardware and software can be attributed to the early state of this technology. We intended to provide the experience of a circuit or a system design cycle, except for layout and fabrication exposure, using programmable and configurable hardware. The user interface for students was developed in Scilab, which called other custom and open source software. Since FPAAs do not have the limitations of pre-fabricated application specific integrated circuits (ASICs), students had the ability to create their own circuit and system ideas continually because the FPAA platform is configurable and programmable. All the groups were able to interact with the FPAA boards and get results, which spoke to the theory that individuals unfamiliar with FPAAs would reap benefits of having access to these low-power signal processing resources. We are confident that the outcome of our assessment suggests

to continue refining our approach as there is opportunity in the growth of this technology.

Our results show that lecture dominance in a course can be decreased by adopting portable, low-cost experiment modules, which is also consistent with researchers in this area [5, 47]. We are not alone in thinking that students need experience taking measurements from hardware and we should provide them files with their results for further data analysis [13, 15]. We believe that a remote testing infrastructure gives students more flexibility to complete labs [37, 45, 73]. Similar to [63], our aim is to improve student learning by incorporating pedagogical methods such as learning in groups, through projects, and by doing experiments. Designing circuits to be manufactured in silicon and collecting data within a single semester is no easy feat [61]. Allowing students to test their designs in silicon helps solidify understanding of circuit concepts. The turn around fabrication time does not fit within a semester because the students have to learn the material, create a design, simulate for varying conditions, and then send their designs to the foundry to be manufactured. Testing, where most of the learning occurs, would not be a part of the curriculum in the scenario described.

The maturation of our pioneering research has inspired our infrastructure's use in this course to augment instruction of analog concepts. In our opinion advocating the unity of innovative research and creative teaching bolsters progressive momentum in both. Thus, an eminent result of appreciating contemporary teaching techniques and concepts is the continuous improvement of FPAAs and RASP Tools. With an enhanced framework, we could explore intricate group dynamics of a team for an engineering project by emulating the methods described by [40]. Their assessment process was pre- and post-course surveys, weekly activity logs, and post-course semi-structured interviews. They analyzed correlation of self-efficacy, gender, and learning goals to task choices in a group setting. They found that goal setting had the desired effect in reducing gendered task choices as also proposed in [52]. A correlation between

time spent on tasks to pre-project learning goals was also discovered [88]. Other researchers highlight the need for active engagement tools in a flipped or inverted class setting, which in his case was the use of an iClicker [74]. He concluded that the use of active tools helped students become active learners, with increased interaction with both their supervisor and peers.

We should stress that the infrastructure for this class is in the development phase, where the hardware and software are improved each semester which affects the teaching approach. This course has always involved an experimental piece that has evolved from pre-fabricated boards for each project and multiple software programs to the versatile FPAA and centralized software. Thus, it is hard to have a control case where experiments and class structure are the same every year. In fact we may have to consider outside factors that affected our course: other classes taken simultaneously, career work, and internships. Our tools are open source, not highly specific, and can be modified to be used with other ICs. This allows collaborators to explore our work and contribute because they have access to boards through our remote system to take experimental data. The base program Scilab and it sub-environment Xcos use flow graph to design, which is similar to MATLAB and Simulink. Students in this class will be familiar with analyzing data in a base program and manipulating parameters in its sub-environment. This tool experience is valuable for future projects that the students will do. To conclude, we emphasize that a blended teaching approach for circuits courses is very important and can be scaled to other electrical engineering courses as well.

CHAPTER V

CONCLUSION

This dissertation provides contributions with respect to emphasizing the use of FPAA SoCs and RASP Tools in analog education. This is a foundation to develop and nourish a balanced design community, where the same resources will be useful for individuals interested in research and prototyping applications.

5.1 Research Brief: Mixed-Signal Tool Suite and Analog Education

In Chapter 1, the dilemma that the mixed-signal design community has a disproportionate ratio of analog to digital designers was presented. I attributed this occurrence to the rise and acceptance of FPGAs and its complementary toolset. Though analog circuits and systems have positive characteristics that would contribute to an overall reduction of power consumption, students are not choosing to take advantage of analog attributes. I hypothesized the education system is not producing an equipped and eager workforce that will be effective in the field because student knowledge reinforcement through experimentation does not often take place. I discussed an idea to use an equivalent device and its software in aligning courses to aid the process of encouraging more engineers to pursue analog design by increasing their familiarity with observable phenomena to deepen their intuition.

In Chapter 2, an open source tool suite that enables simulation of circuits and systems and programming of these designs to configurable hardware was discussed. The details of the software and hardware, RASP Tools and FPAA SoCs, were presented. Similarly, a few examples demonstrating the combined capabilities of the toolset in conjunction with the FPAA were shown.

In Chapter 3, I talked about immersing RASP Tools and RASP 3.0a FPAA SoCs into an analog course. FPAAs and tools were designed to afford students the chance to explore and become engrossed with class material. The inspiration for the set of boards developed for the class as well as the features and utility functions added to the toolset was described.

In Chapter 4, the results of student answered questions on pre- and post-surveys about their experience being in a course that used configurable hardware and it accompanying tool suite was reviewed. I explained how the survey questions enabled students to evaluate their prior knowledge and skills, the learning techniques implemented in the course, and the hardware and software used throughout the course. It was concluded that the assessment results were promising and this research is headed in the right direction.

5.2 Contributions: Design Automation Tool Suite and Course Immersion

I have made research advancements towards the goals of creating a foundational tool suite, which complements the versatility of programmable and configurable mixed-signal FPAA SoCs, and highlighting the impact of analog circuit and system design utilizing these resources in analog education. These accomplishments were achieved with the assistance of several members of the Integrated Computational Electronics (ICE) Laboratory as noted below. The work listed includes custom tools, user interfaces, scripts, algorithms, a printed circuit board (PCB), and an evaluation of the aforementioned entities.

- Co-designed the high-level tools infrastructure called **x2c** → Xcos to Chip with Suma George, which translates a Xcos design to a switch list that is for programming an IC. Led **sci2blif** module that converts Xcos design to BLIF, which also gives users the option to individually choose the placement location of their

design components within the FPAA SoC array. Example Xcos files were made for demonstrating features and providing users working systems that serve as a starting reference to build their own designs. Developed RASP Tools, an open source environment, within an Ubuntu VM. Components of RASP Tools include the launching icon, a main RASP Tools GUI, other GUIs (voltage measurement, design metrics, and calibration), utility functions (create and load csv file into Scilab, initialize ammeter equipment for current measurement, *etc.*), updating features (allow students and collaborators to receive an updated RASP Tools version), FPAA and user populated palettes in Xcos, and macromodels for simulation of Level 1 blocks in Xcos [18].

- Extracted co-design metrics from compiled designs to display useful information to designer (power, area, and number of individual components used). Isolated routing capacitance (global and local) of circuit components to determine the corresponding FG Ibias value to satisfy the user specified frequency parameter.

- Led the design of a PCB board for RASP 3.0a FPAA SoC with the aid of Stephen Nease and Farhan Adil. Then created technology files for this chip along with Suma George, which describes the arrangement and architecture of the CABs and CLBs. These files are called by the **x2c** tool and specifically *vpr2swcs* [17].

- Co-taught graduate course ECE 6435 *Neuromorphic Analog VLSI Circuits* along with Sihwan Kim in Spring 2015. A virtual machine containing RASP Tools was used in conjunction with RASP 3.0a FPAA SoC boards mentioned above [17,36].

- Built the remote system front-end that was integrated into the RASP Tools infrastructure with Ishan Lal to give students the ability to take experimental results without a local board [37, 73].

- Assessed the students' satisfaction of using RASP Tools and FPAA SoCs in ECE 6435 for the Spring 2016 semester via surveys. The evaluation of the teaching approach as well as the self-reported confidence of the students' skills and knowledge of course material were gathered [19].

APPENDIX A

BLOCK FILES: ODE SIMULATION

Typically, FPAA blocks are type 5 because the code is written in the Scilab language and not the C language. The interfacing function utilizes a scicos_model data structure that has fields such as inputs, outputs, block parameters, states, *etc* ($model.field$) [70]. The computational function, employs a sci_struct data structure that has similar fields ($block.field$) [72]. The interfacing function defines block and user parameters (*e.g.*, $model.ipar$, $model.rpar$, and $model.opar$) and stores them in its data structure. These parameters are shared and can be extracted in the computational function via its own data structure.

Simulation of a block is dependent on the computational function. There are ten flags that can be set to dictate an action during the simulation process. FPAA blocks usually use two of the flags. First, flag = 0, computes the derivative of the continuous-time state ($block.xd$); where the state field ($model.state$) contains a vector of the initial values of continuous-time state ($block.x$). ODEs are written in this section. Second, flag = 1, calculates the outputs of the block ($block.outptr$). The continuous-time state ($block.x$) and user parameters (*e.g.*, $block.ipar$, $block.rpar$, and $block.opar$) can be used in the equations of these flags. Also, the current simulation time function $scicos\_time()$ is accessible to these flags.

FPAA blocks support vectorization. When the interfacing function is created the parameters may be saved in a vector or matrix format (parameters separated by commas or semicolons, respectively).

For Example: a block representing three circuits (*i.e.*, $1, 2, 3$) and each circuit has

three parameters (*i.e.*, a, b, c).

$$\begin{bmatrix} a_1 & a_2 & a_3, & b_1 & b_2 & b_3, & c_1 & c_2 & c_3 \end{bmatrix}$$

$$\begin{bmatrix} a_1 & a_2 & a_3; & b_1 & b_2 & b_3; & c_1 & c_2 & c_3 \end{bmatrix} \leftrightarrow \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}$$

The equations for indexing parameter values are written such that all values of a parameter are accessed at once using vector notation [*e.g.*, (c_1, c_2, c_3)].

Scilab's indexing of vector and matrix data structures are:

$$\begin{bmatrix} 1 & 2 & 3, & 4 & 5 & 6, & 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

The equations work for both vector and matrix data structures. However, a matrix must be transposed to comply with Scilab's indexing; otherwise, all parameter values of a circuit would be read together [*e.g.*, $[a_2, b_2, c_2)$] instead of a single parameter for each circuit (*i.e.*, vectorized parameter) [*e.g.*, (b_1, b_2, b_3)].

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}^T \longrightarrow \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}$$

We need a start and end position $\rightarrow block.field(Start : End)$

Where P_N - parameter number (*i.e.*, $a = 1, b = 2, c = 3, etc$) and N_B - number of circuits represented by block (*e.g.*, $[1, 2, 3] = 3$).

$$Start = (P_N - 1) \times N_B + 1 \tag{5}$$

$$End = P_N \times N_B \tag{6}$$

As an aside, a different equation is needed to access parameter values during compilation to BLIF. Parameters are not indexed using vector notation because BLIF is written for a circuit instance one at a time (*i.e.*, only parameters for that instance are needed).

Where T_P - total number of distinct parameters (*e.g.*, $[a, b, c] = 3$) C_I - circuit instance of block (*i.e.*, $1, 2, 3,$ *etc*) P_N - parameter number (*i.e.*, $a = 1, b = 2, c = 3$, *etc*).

$$T_P \times C_I - (T_P - P_N) \tag{7}$$

APPENDIX B

PSEUDO CODE OF ALGORITHMS

To develop the **x2c** tool many algorithms and scripts were written. The pseudo code of only five algorithms are disclosed in this appendix. One of two extensive parts of the **x2c** tool is the overarching program **sci2blif** that handles compilation of user designs to BLIF (netlist). The pseudo code of *vpr2swcs* that is the other half of the **x2c** tool is not discussed. The four other algorithms (*i.e.*, netlist population, user placement, global capacitance, and routing capacitance) under the umbrella algorithm **sci2blif** are described. Thus, **sci2blif** contains these sub-algorithms and calls other scripts not detailed explicitly below to accomplish its compilation task.

sci2blif is a function that is called by the user when they press the *Compile Design* button on the main GUI. It is actually called twice before a final switch list is produced. The first call extracts global routing capacitance and the second computes the actual Ibias values for FGs to be targeted during programming of FPAA hardware; the Ibias information is passed from the netlist to the switch list. Before building and utilizing **x2c** for compilation, developing the comprehension of Scilab/Xcos and how to make FPAA specific blocks was important. This ground work led to the discovery of the *scs_m* data structure that is essentially the blueprint of the user's design, which contains the circuit or system elements and their parameters. The process to generate a BLIF file is dependent on information within the *scs_m* data structure.

The netlist population algorithm both manages the netlist matrix and routing capacitance matrix. This procedure operates on the fact that block and link *data* objects of *scs_m* are enumerated in what seems to be a non-deterministic manner, but actually reflects the manipulations of the user within the Xcos environment.

Similarly, this procedure assumes that when parsing the link *data* the link object going to a Split_f block will always be encountered before the link object coming from the Split_f block. Thus, a corresponding net-name number will be known and can be placed in the netlist matrix for the block coming after the Split_f block. Capacitance for a block is comprised of global and local routing and is saved in a separate matrix. Part of the local capacitance of a block is in fact the input capacitance of the block connected to its output port. Since the user's design can also contain inherent Scilab blocks along with FPAA blocks (both Xcos and Modelica types), they are ignored when totaling this input capacitance. The remaining input capacitance of a block is determined outside this netlist population algorithm; it is the summation of internal capacitance in the block's circuit affecting its output.

The placement algorithm updates the *place* file to use locations specified by the user for circuit elements in their design. The global capacitance algorithm is called within *vpr2swcs*. It tallies the routing capacitance along the path through C and S blocks from an output to the input for each block and saves these values in a file. The routing capacitance algorithm imports these global capacitance values and makes them available for **sci2blif** to evaluate the total routing capacitance.

Algorithm 1 Pseudo code for **sci2blif**

```
1: function SCI2BLIF(iteration_num, routing_cap_info)
2:                                          ▷ sci2blif is first part of x2c tool
3:                                  ▷ scs_m: data structure for a Xcos diagram
4:                                      ▷ scs_m contains block and link objects
5:     Declare and set variables
6:     for each object i ∈ scs_m do
7:         Determine if current object is a block or link
8:         Update count for number of blocks
9:         Update count for number of links
10:        Increase count for particular type of block: input, output, etc
11:        Set flags for special conditions
12:        Update max number of inputs for a block
13:        Update max number of outputs for a block
14:                        ▷ Matrix supports block with max inputs and max outputs
15:    Create empty netlist matrix
16:            ▷ Each link object has to and from (start and end) block information
17:    Populate netlist matrix              ▷ See Algorithm 2 - Netlist Population
18:                                                ▷ Create BLIF file for design
19:    Write input and output net names in BLIF file
20:    for each object z ∈ blocks do
21:        if Current block = input block then
22:            Write location of input to pads file
23:        else if Current block = output block then
24:            Write location of output to pads file
25:        else if Current block is digital and uses verilog code then
26:            Create secondary BLIF file
27:            Run VTR
28:            Concatenate generated BLIF to overall design BLIF file
29:        else
30:            Match current block to corresponding BLIF block module
31:            if iteration_num = 2 then
32:                Retrieve routing_cap_info
33:                Compute total routing cap   ▷ global, local, and input capacitance
34:                                                ▷ Floating Gate (FG) transistor
35:            Compute Ibias values for FGs using total capacitance
36:            Update total number of elements utilized   ▷ For Design Metrics GUI
37:            Write netlist code in BLIF file
38:    Rename net names if flagged
39:                        ▷ User can specify tile location of CAB and CLB elements
40:    Perform user placement if flagged      ▷ See Algorithm 3 - User Placement
41:    Run vpr2swcs to obtain switch list                ▷ Use correct technology file
42:    ▷ Generate global routing cap file → See Algorithm 4 - Global Capacitance
43:    Import capacitance values       ▷ See Algorithm 5 - Routing Capacitance
44:    return routing_cap_info
```

Algorithm 2 Pseudo code for Netlist Population

```
1: procedure NETLIST POPULATION
2:     for each object j ∈ links do
3:         Retrieve from block information
4:         for each object k ∈ blocks do
5:             Find block that matches current from block
6:             Determine indexing to place net number within netlist matrix
7:             if Block = Split_f then                    ▷ Scilab inserted node block
8:                 Retrieve net number associated with Spilt_f block instance
9:                 Update netlist matrix
10:                if iteration_num = 1 then              ▷ iteration_num is either 1 or 2
11:                    if Current from block is a FPAA block then
12:                        Update routing_cap_info to reflect input cap of to block
13:                    else if Current from block is a Modelica block then
14:                        if Modelica block is FPAA designated then
15:                            Update routing_cap_info to reflect input cap of to block
16:            else if Block = Join then                  ▷ Node block for ≥ 2 connections
17:                Update netlist matrix
18:                if iteration_num = 1 then
19:                    if Current from block is a FPAA block then
20:                        Update routing_cap_info to reflect input cap of to block
21:                    else if Current from block is a Modelica block then
22:                        if Modelica block is FPAA designated then
23:                            Update routing_cap_info to reflect input cap of to block
24:            else                                       ▷ FPAA block that compiles to CAB or CLB
25:                Update netlist matrix
26:                if iteration_num = 1 then
27:                    Add net name to routing_cap_info
28:                    Update routing_cap_info using net name of from block output
29:            break
30:        Retrieve to block information
31:        for each object g ∈ blocks do
32:            Find block that matches current to block
33:            Determine indexing to place net number within netlist matrix
34:            if Block = Split_f then
35:                Save net number associated with Spilt_f block instance
36:                if iteration_num = 1 then
37:                    Initialize routing_cap_info for input capacitance of to block
38:            else if Block = Join then
39:                if iteration_num = 1 then
40:                    Initialize routing_cap_info for input capacitance of to block
41:            else                                       ▷ Regular FPAA block
42:                Initialize routing_cap_info for input capacitance of to block
43:            Update netlist matrix and net number
44:            break
```

Algorithm 3 Pseudo code for User Placement

```
1: procedure User Placement
2:     ▷ User can specify the location of circuit or system elements in FPAA fabric
3:     Run VPR to obtain initial place file                    ▷ Use correct technology file
4:       ▷ place file → block name, [x, y] coordinates, sub-block, and block number
5:             ▷ block name (blk_name) is equivalent to net name of block output
6:     Open, parse, and save place file contents
7:     for each object q ∈ non-header portion of file do
8:         Extract and save [x,y] coordinates and sub-block (hereby [x,y])
9:     old_locations ← reconstructed spacing of blk_name and [x,y]
10:    for each object v ∈ new_locations do
11:        for each object r ∈ old_locations do
12:                                  ▷ Locate desired blk_name that will get user [x, y]
13:            if blk_name in old_locations(r) = blk_name in new_locations(v) then
14:                for each object p ∈ old_locations do
15:                                                ▷ Locate blk_name with user [x, y]
16:                    if [x, y] in old_locations(p) = [x, y] in new_locations(v) then
17:                                              ▷ Replace user [x, y] with non-user [x,y]
18:                        [x, y] in old_locations(p) ← [x, y] in old_locations(r)
19:                                            ▷ Update desired blk_name with user [x,y]
20:                [x, y] in old_locations(r) ← [x, y] in new_locations(v)
21:                                                        ▷ Create new place file
22:    Format old_locations (blk_name and [x,y])
23:    Write header information in place file
24:    Write formatted blk_name and [x,y] in place file
```

Algorithm 4 Pseudo code for Global Capacitance

```
1: procedure GLOBAL CAPACITANCE
2:                                   ▷ Occurs within the vpr2swcs algorithm
3:         ▷ Computes the global routing cap (*) for each block and saves in a file
4:     ▷ (*) along path through connection (C) and switch (S) blocks to destination
5:                                   ▷ C and S blocks are inside CABs and CLBs
6:     Declare and set variables to store net names and cap values
7:                                   ▷ Within global C block switch generation module
8:                                   ▷ Save new net name info
9:     if Last net name in list ≠ current net name then
10:        Append net name to list
11:        Append C block cap value to a separate list with the same index
12:                    ▷ Update routing capacitance if C block is along path again
13:    else if Last net name in list = current net name & no feedback then
14:        Increment capacitance with C block value
15:    Update global routing sequence with C block marker
16:                    ▷ Within global switch S block switch generation module
17:                                  ▷ Save new net name info
18:    if Last net name in list ≠ current net name then
19:        Append net name to list
20:        Append S block cap value to a separate list with the same index
21:                    ▷ Update routing capacitance if S block is along path again
22:    else
23:        Increment capacitance with S block value
24:                                  ▷ Account for C block in between two S blocks
25:    if If previous block marker in routing sequence = S block then
26:        Increment capacitance with C block value
27:    Update global routing sequence with S block marker
28:    ▷ New module: create file with net names of block outputs and its global cap
29:    Remove dummy net name and capacitance value
30:    Open a blank file
31:    for each object t ∈ net names variable do
32:        Write net name and total global routing capacitance in file
```

Algorithm 5 Pseudo code for Routing Capacitance

```
1: function ROUTING CAPACITANCE(routing_cap_info)
2:     Import generated file to get global routing capacitance values
3:                                  ▷ file contains net name of output of each block in design
4:     for each row object w ∈ cap file do
5:         if net name is generic then                                        ▷ Ex: netX_Y
6:                                                  ▷ To support vectorization of blocks
7:             Determine base net name                              ▷ netX in above Example
8:             Save base net name
9:             Determine number of net name for indexing          ▷ Y in above Example
10:            Save number of net name for indexing
11:        else                                                               ▷ Custom name
12:            Save custom net name
13:            if Custom net name does not contain a number then
14:                                                    ▷ Ex: some_var or somevar
15:                Save number 1 for indexing
16:            else                                  ▷ Ex: some_var12 or somevar12
17:                Save number in net name for indexing
18:        Update routing_cap_info with capacitance value using index and net name
19:    return routing_cap_info
```

APPENDIX C

VM SETUP AND REMOTE SYSTEM GUIDE

RASP Tools and the FPAA ICs were completely new to the students in the graduate course. Thus, it was imperative to provide the class introductory material to get started. I decided to create a step-by-step guide with associated pictures to best facilitate setup procedures and testing. The guide goes through the process of installing the ubuntu virtual machine (VM) with RASP Tools encapsulated, creating a design, and obtaining experimental measurement data. The goal was to have a comprehensive and straightforward guide that highlighted features and the characteristic processes of RASP Tools to complete desired actions, which could also be used as a reference later on.

There is a dedicated download website for students and other potential users that is referenced [http://users.ece.gatech.edu/phasler/FPAAtool/index.html]. There they can find the link for the VM manager software, VirtualBox, and the VM ova file. Then after launching VirtualBox, the importing directions are given and a valuable tip to create a shared folder between host machine and VM for easy transferring of files. Upon successful importation, the password for the VM is provided and the action to launch RASP Tools. The updating feature is introduced and utilized, which is useful going forward to obtain the latest improvements of the tool suite. A default directory was created where students can place there folders for each personal design or assignment; two ways to create a folder in the ubuntu environment are shown.

Next, the guide presents an assignment that utilizes the remote system for data acquisition to verify the installation and give students experience with the tool suite. Then the guide shows how to create a design with the graphical representations of

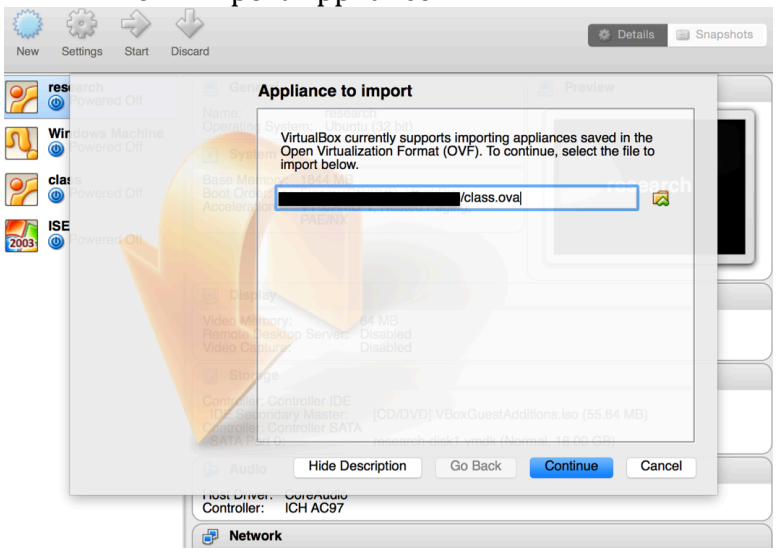
circuit elements, adjust parameters, and setup input voltages. The specific remote system chip number is given and the procedure to compile the design and send the email with the design details are shown; each FPAA chip has a unique identification number to specify the underlying files for programming. Upon receiving an email with the results from the remote system, the guide states how to load these results in the Scilab environment.

Becoming Familiar with the RASP Tools Suite for FPAAs

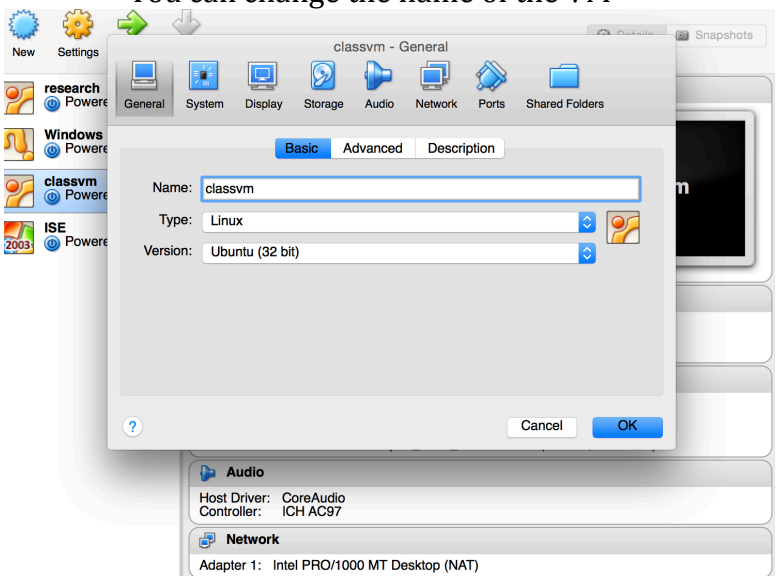
Setup: [Use links on Virtual Machine (VM) Download Website]

- Download/Install Virtual Box Platform Packages 4.3.20 (latest version)
- Download/Install Virtual Box Extension Pack 4.3.20 (latest version)
- Download Virtual Machine: OVA file for Virtual

1. Launch and Import Virtual Machine
   File -> Import Appliance

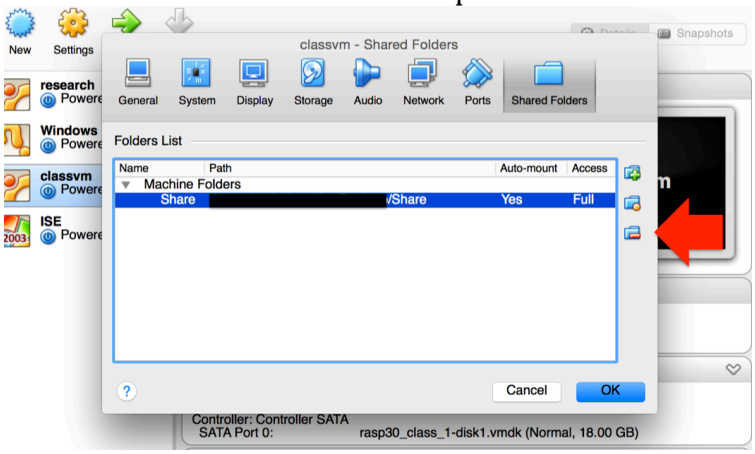


2. Select classvm in left-hand windowpane of Virtual Box and click Settings
   ->You can change the name of the VM



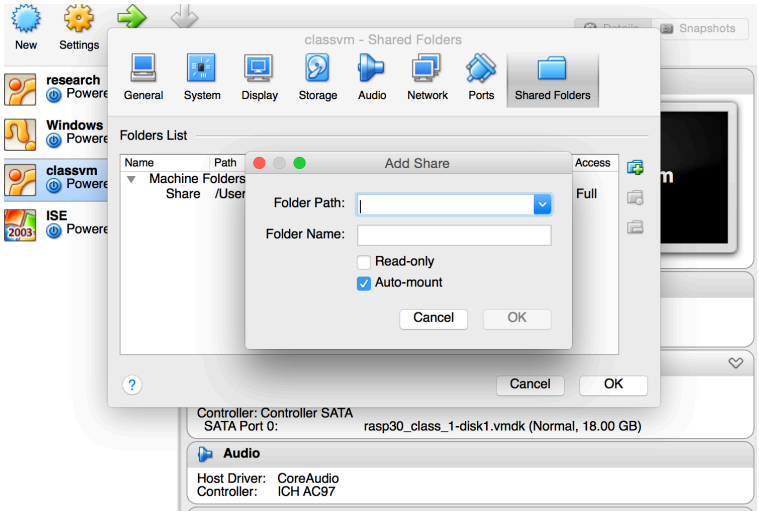
3. Create a Shared Folder [Allows transferring of files from host machine to VM]

➔ First select and delete previous Shared Folder



➔ Make/Select an exiting folder

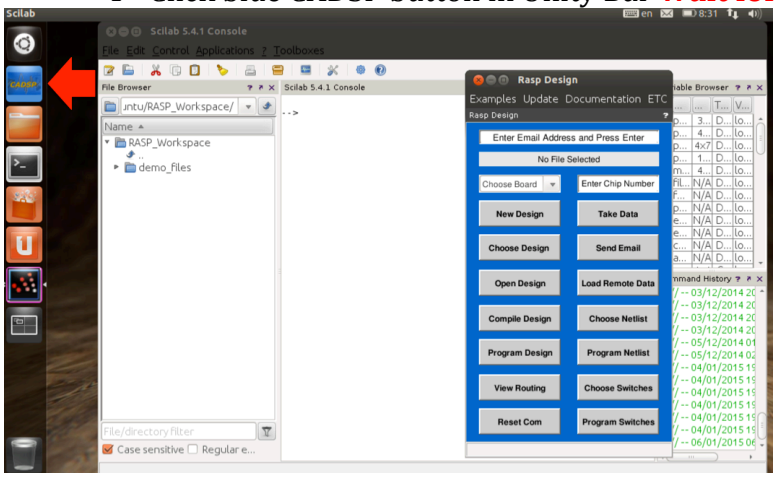
➔ Auto-mount should be selected



4. Select the VM and press the green Start Button next to Settings

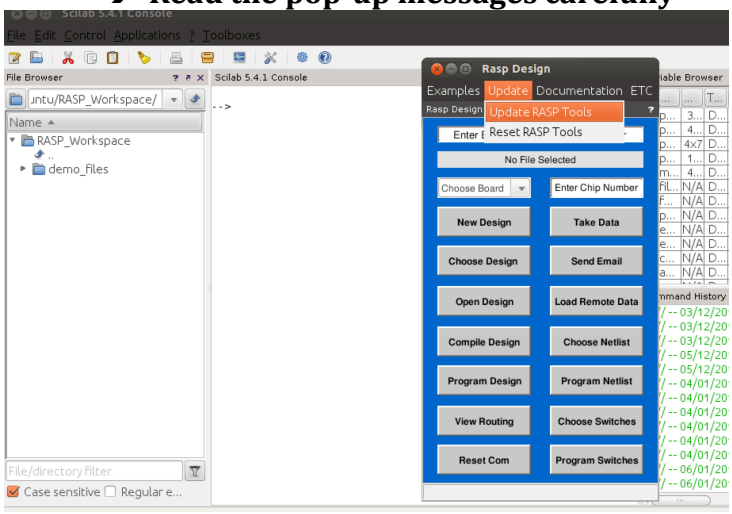
➔ Password is “reverse”

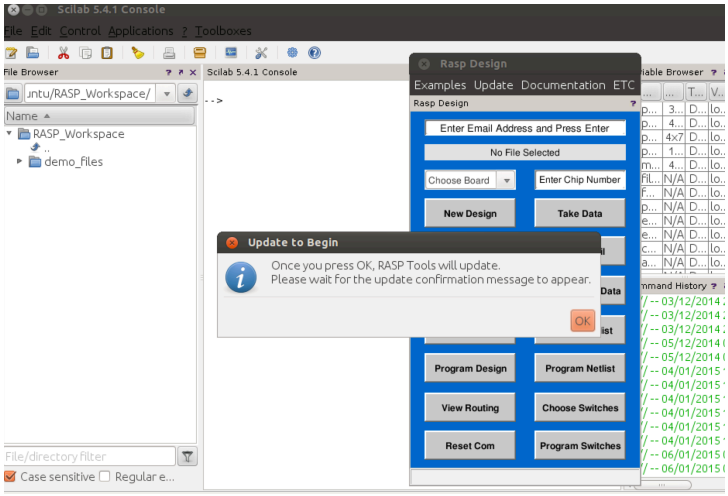
➔ Click blue CADSP button in Unity Bar **Wait for the tools to load**



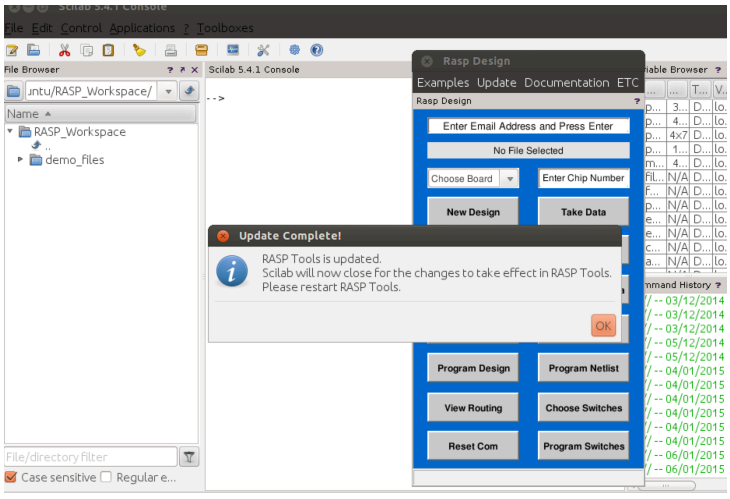
5. Update RASP Tools to the latest version **(You must be connected to the Internet)**

➔ **Read the pop-up messages carefully**





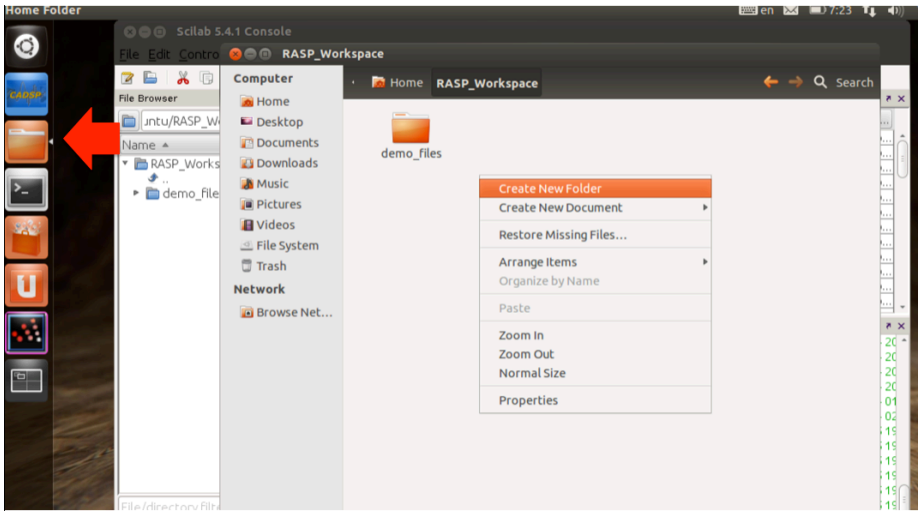
Wait for the Update Complete message! ...Press ok, and then launch the Tools again.



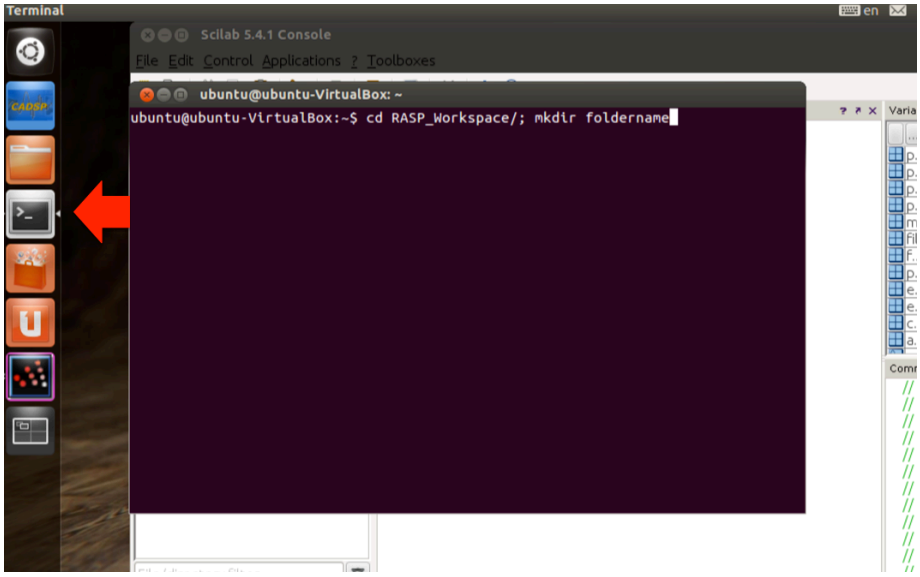
Note:

1. Notice that the default directory is RASP_Workspace
* We encourage you to make a folder for each of your designs to stay organized

2. You can make Folders different ways **(Create a Folder for your assignment)**
    - Folder Icon ➔ Navigate to RASP_Workspace and right click for menu



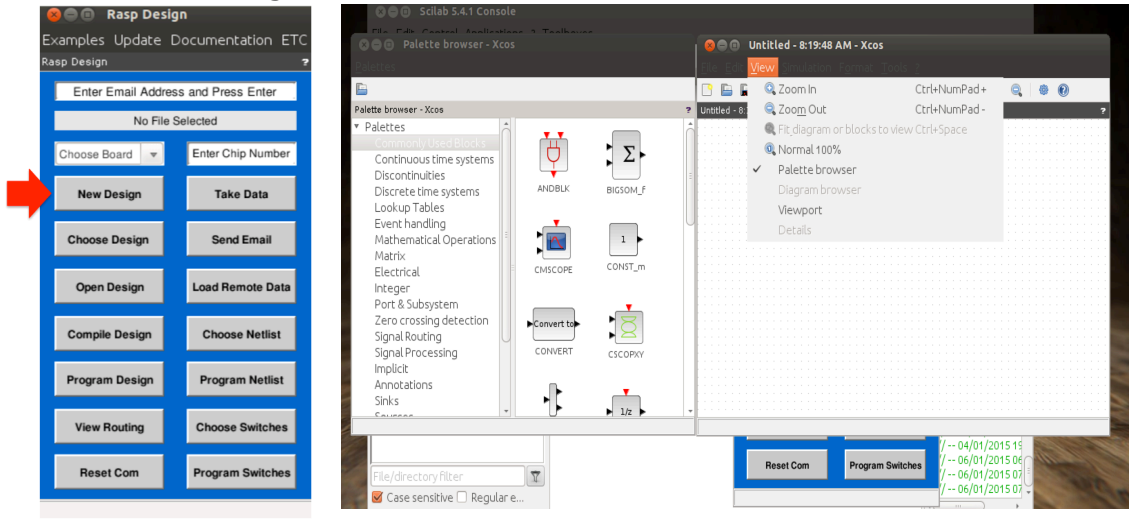
➔ Terminal



Assignment

Take measurements using our remote system.

1. Create a New Design

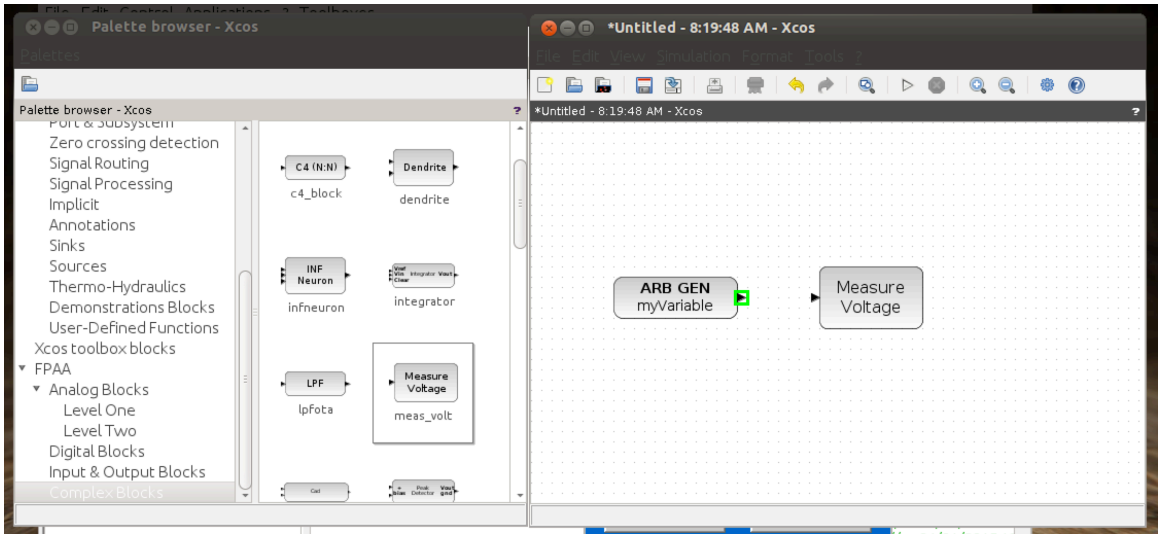


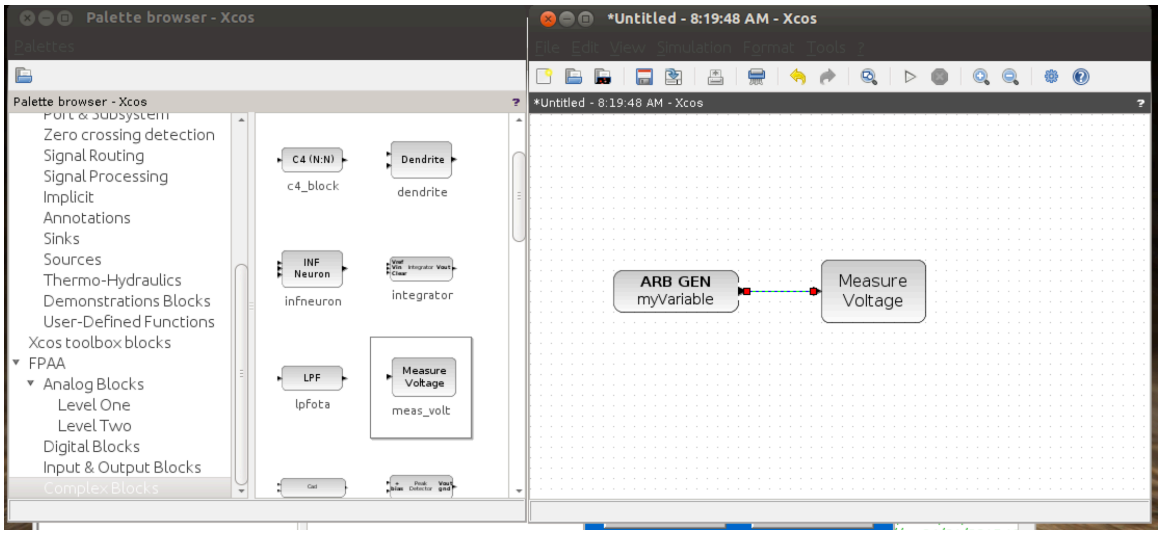
Note:

- Palette Browser contains Scilab standard blocks
- Palette Brower contains RASP Tools library of blocks
    - Scroll down to view Palette “FPAA”
- Palette Browser can always be opened from View Tab ➔Palette browser

2. Drag and drop blocks to Xcos window

➔ Arbitrary Waveform Generator block (Input & Output Blocks)
➔ Voltage Measurement (Input & Output Blocks)
➔ Click arrow then drag line connector to other arrow

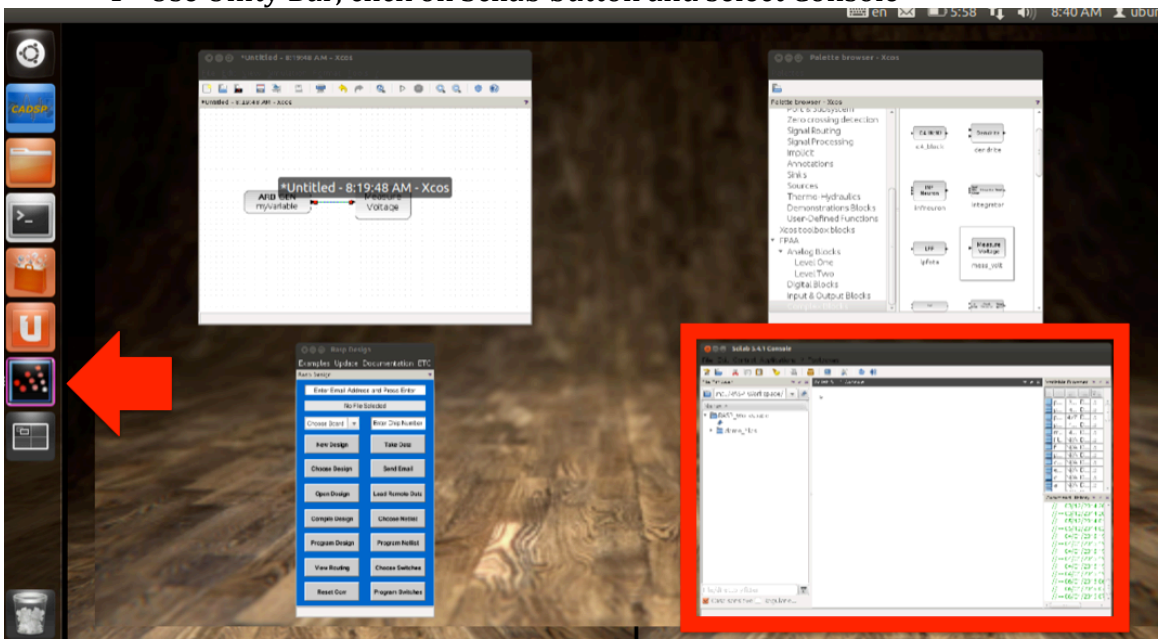




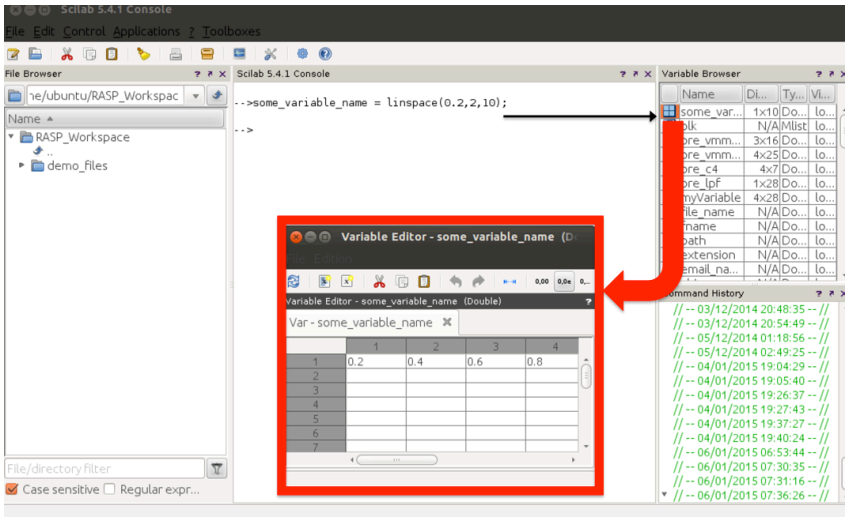
3. Navigate to the Scilab Console

Either

- ➔ Alt+Tab (keep pressing tab until you highlight “Scilab 5.4.1 Console”)
- ➔ Use Unity Bar, click on Scilab button and select Console



4. Create a variable containing a vector of voltage values (Min: 0.2 V, Max: 2.5V)
   - ➔ Using a ";" after a command will not display the result in the console
   - ➔ After pressing Enter, your variable will appear in the Variable Browser
   - ➔ By double clicking on the variable, the Variable Editor will appear

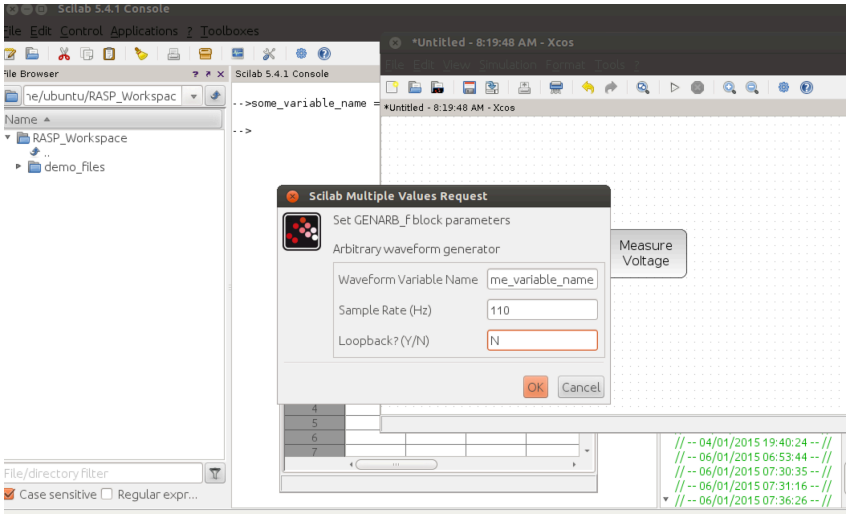


*Help on linspace:

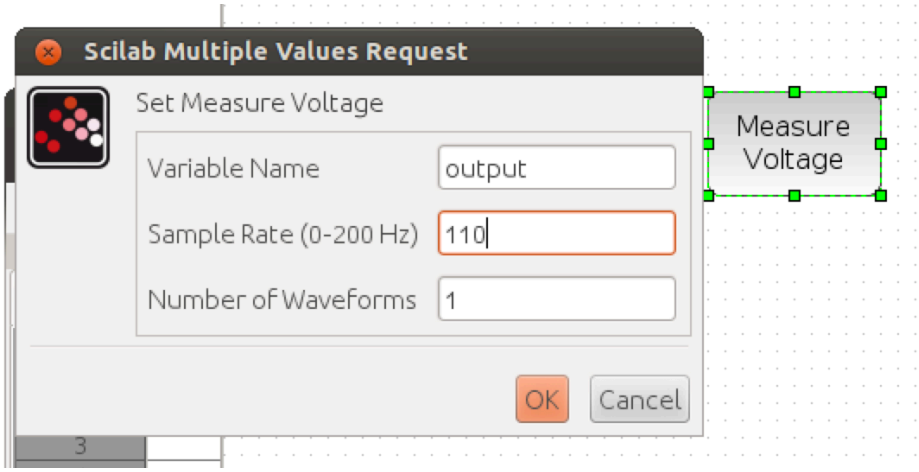
`[v]=linspace(x1,x2 [,n])`

Linearly spaced vector. linspace(x1, x2) generates a row vector of n (default value=100) linearly equally spaced points between x1 and x2.

5. Navigate to the Xcos window and change Arbitrary Generator block's parameters
   - ➔ Choose One:
     - Double click the block • Right click on the block and select "Block Parameters" • Press Ctrl+B
   - ➔ Change all three parameters and press OK
     - Type in the variable you created
     - Use a Sample Rate < 200
     - Change Loopback to "N"

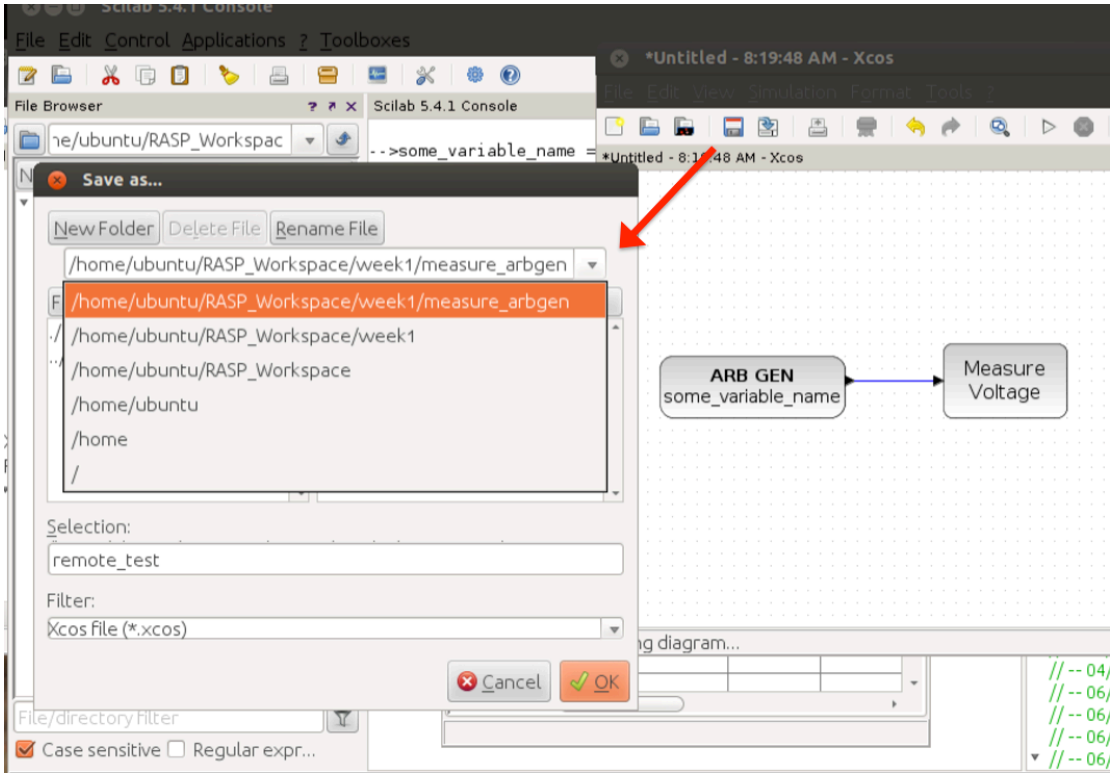


6. Change Measure Voltage block's Sample Rate parameter to the value you chose for Arbitrary Waveform Generator and press OK

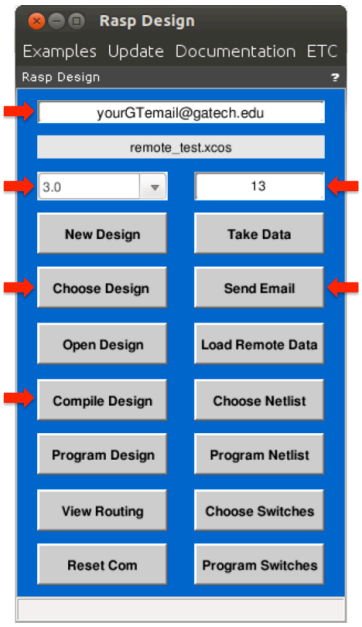


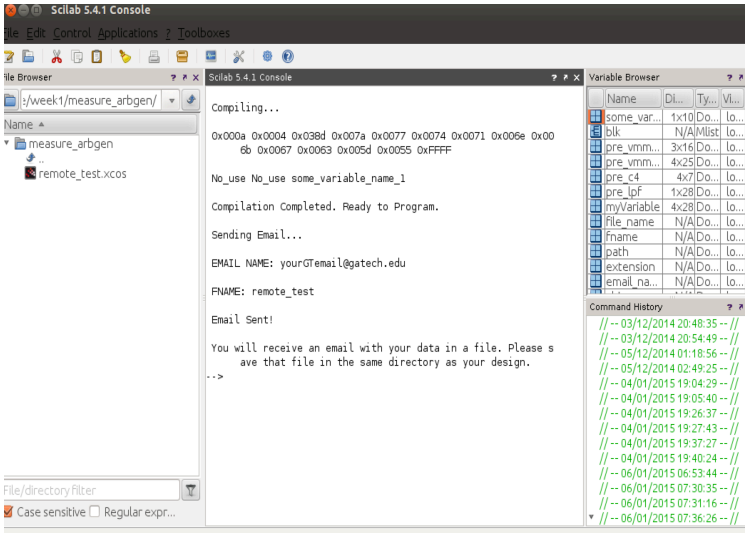
7. Save your Design

- Use File ➔ Save as… OR Save icon in the toolbar
- Navigate to the folder you created and under Filter choose "Xcos file (*.xcos)"



8. Go to the main blue GUI (Rasp Design)
    - Type your GT email address (Press Enter)
    - Type “13” for chip number (Press Enter)
    - Choose 3.0 Board from drop-down menu
    - Choose your design
    - Compile your design
    - Send email





*You will receive an email with your results from the remote system

9. Load your results into Scilab
    - Rasp Design GUI
        - Choose your design name
        - Type chip number (Press Enter)
    - Load Remote Data
        - Download and Save results.zip attachment to your Shared Folder on host machine
        - Move zip file to your design folder (Use File icon or Terminal)
        - Press Load Remote Data on GUI

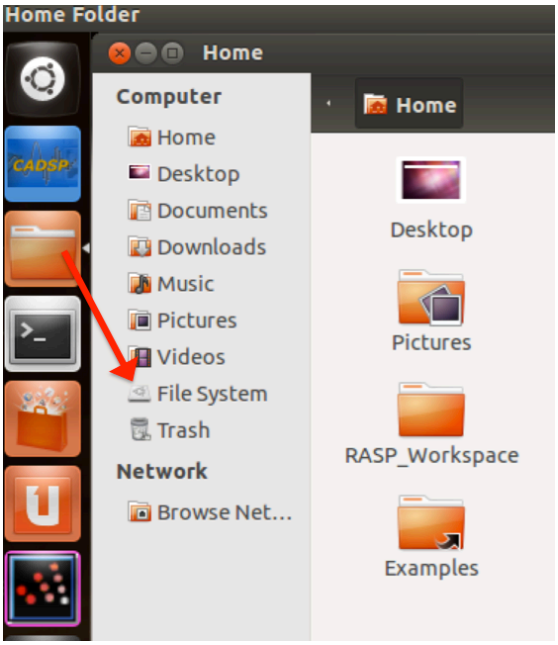
```
ubuntu@ubuntu-VirtualBox: ~
ubuntu@ubuntu-VirtualBox:~$ mv /media/sf_<Shared Folder Name>/results.zip /home/ubunt
u/RASP_Workspace/<design folder location>
```

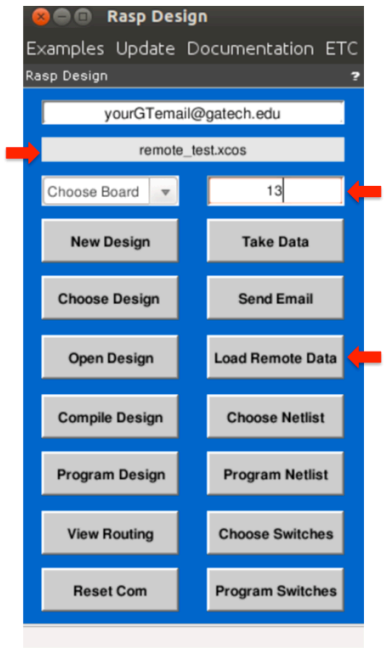
For Example,

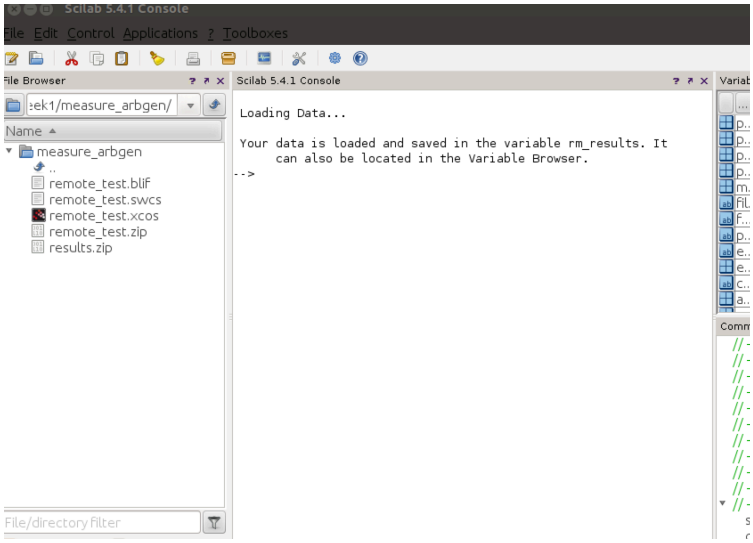
```
ubuntu@ubuntu-VirtualBox: ~
ubuntu@ubuntu-VirtualBox:~$ mv /media/sf_Share/results.zip /home/ubuntu/RASP_Workspac
e/week1/measure_arbgen/
ubuntu@ubuntu-VirtualBox:~$
```

If using File icon to relocate zip file…

- File System ➔ media folder ➔ sf_<Your Shared Folder Name>







10. Look at your results via the Console

Either Type

➔ rm_results (results appear in Console)

➔ editvar rm_results (results appear in Variable Editor)

APPENDIX D

RASP 3.0A PCB PLATFORM SCHEMATICS

The design of the RASP 3.0a PCB is a collection of new additions, previously tested modules, and improvements. It is a compact double sided board designed with a RASP IC and control circuitry combined, whereas the RASP 3.0 PCB is actually two separate boards. The VGA OV7670 camera module with a 18-pin configuration and a 30 frames per second (fps) rate was chosen for this board; pins were routed to the on-chip microprocessor to support I2C communication and directly to the internal FPAA fabric for data manipulation. Several corrections from earlier PCB designs required altering and creating footprints for layout (*i.e.*, micro USB, inductor, crystal oscillator, *etc*). A push button to activate a reset of the microprocessor or FTDI chip that is dependent on jumper placement was rectified. The audio module was incorporated from a previous board after determining the polarized capacitor connections from an actual old PCB to prevent explosion. Unnecessary device components were removed and routing was corrected (*i.e.*, shorts and opens) after taking into account hassles from calibrating the RASP 3.0 SoC. There is a dedicated download website to get other associated files [http://users.ece.gatech.edu/phasler/PCboards/index.html].

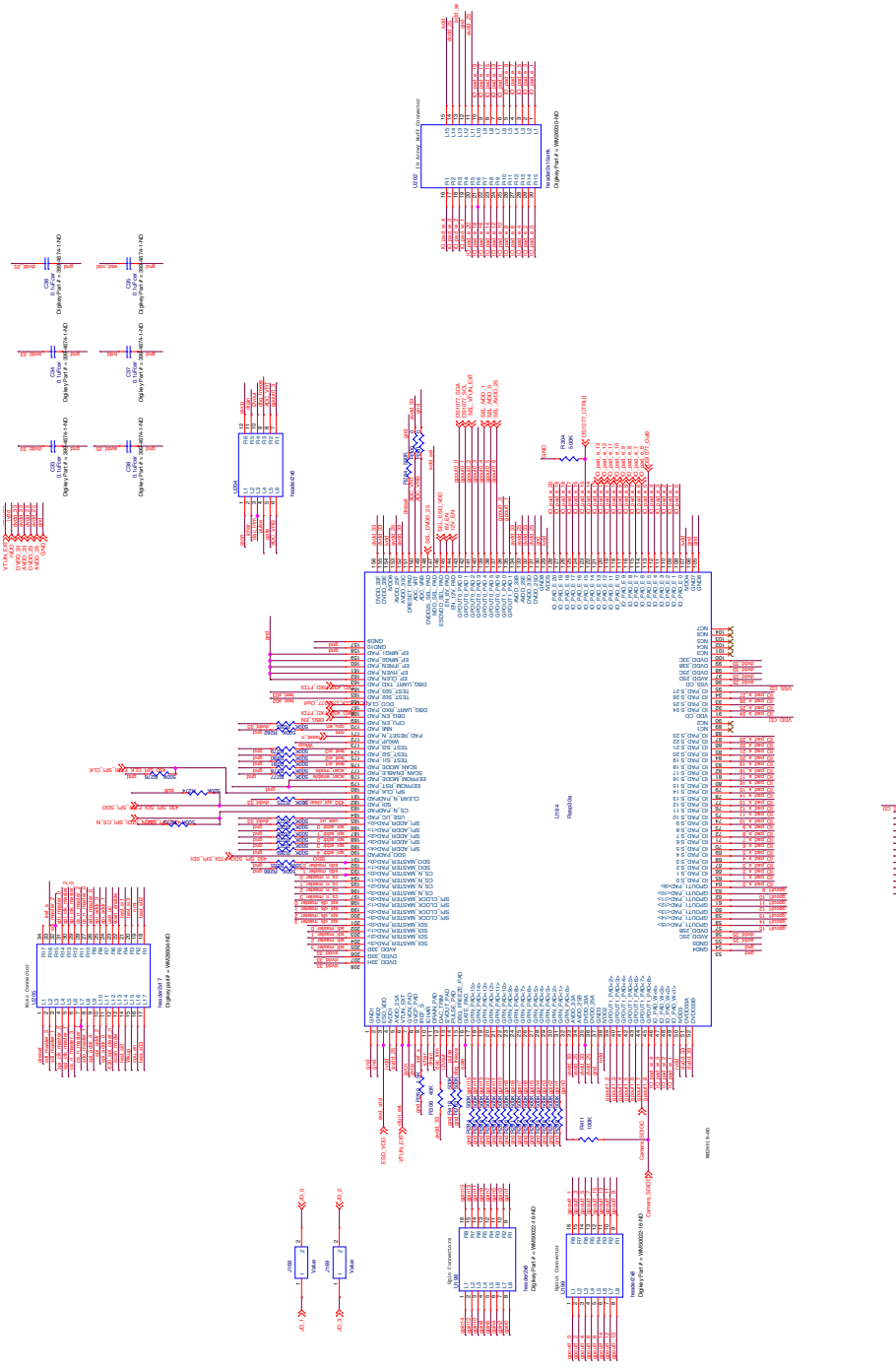


Figure 29: RASP portion of the RASP 3.0a PCB

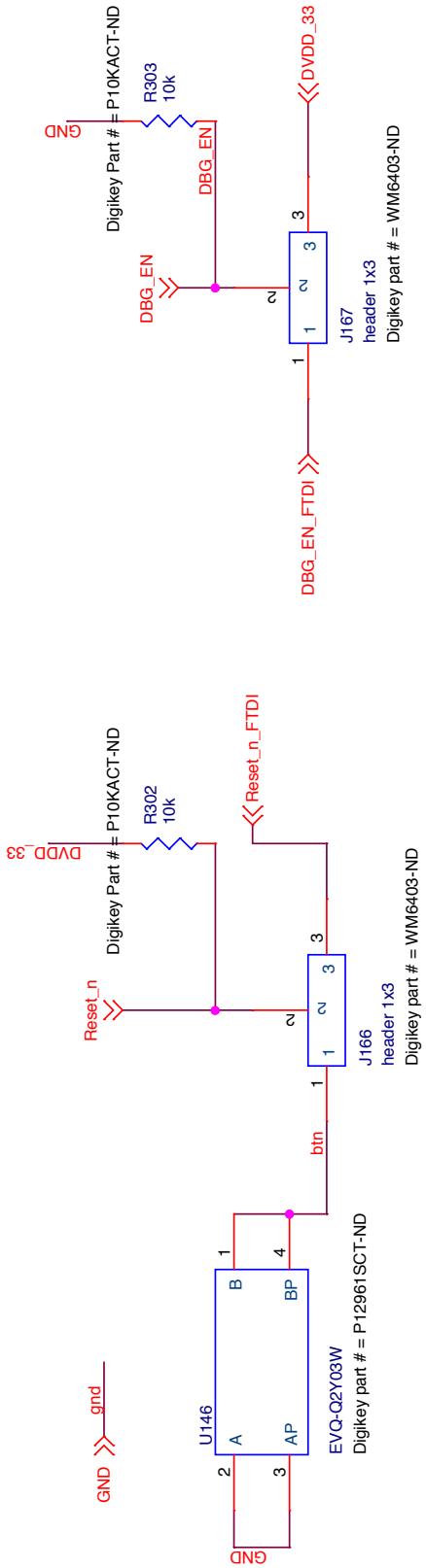


Figure 30: Startup Circuitry portion of the RASP 3.0a PCB

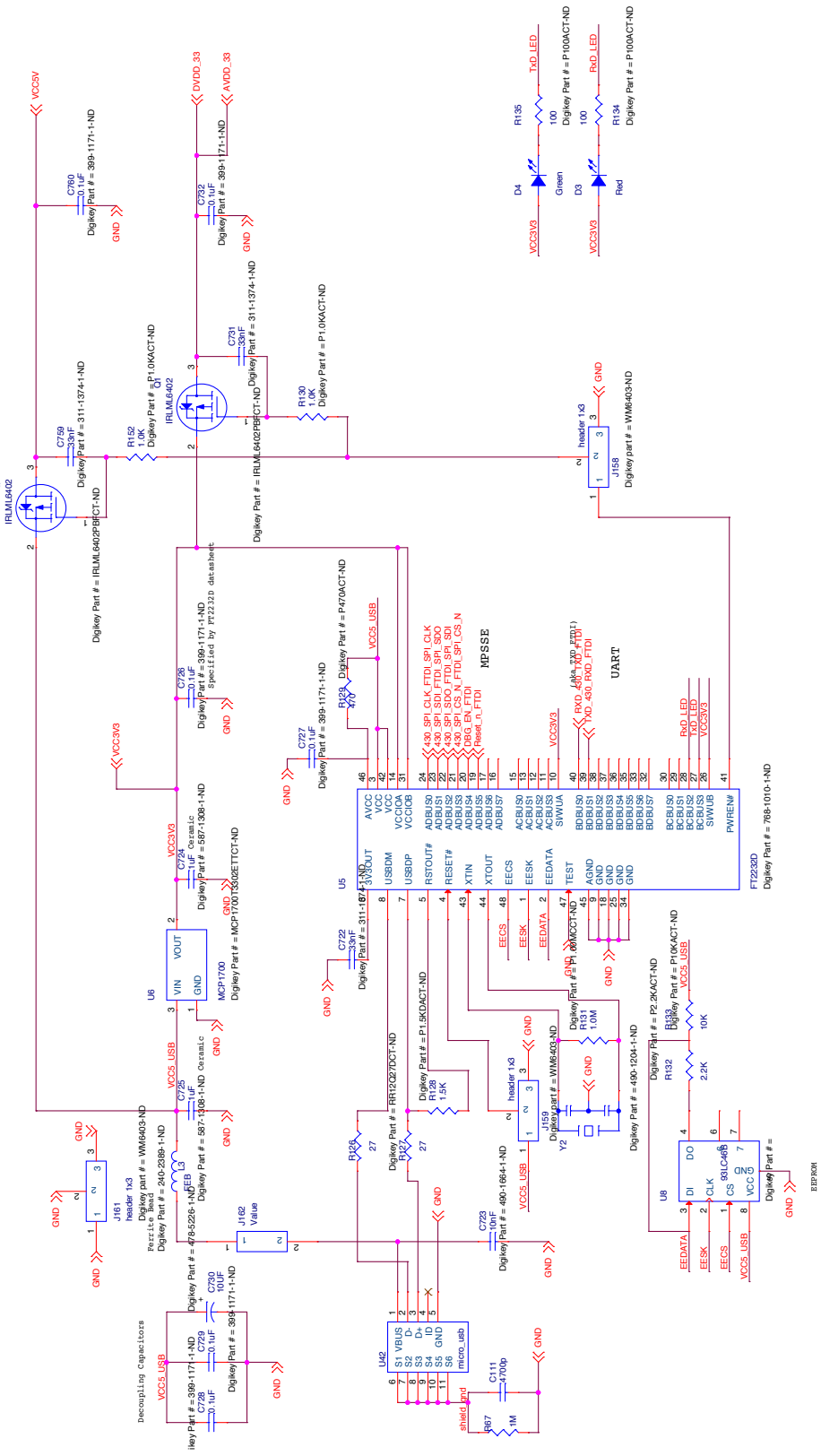


Figure 31: FTDI portion of the RASP 3.0a PCB

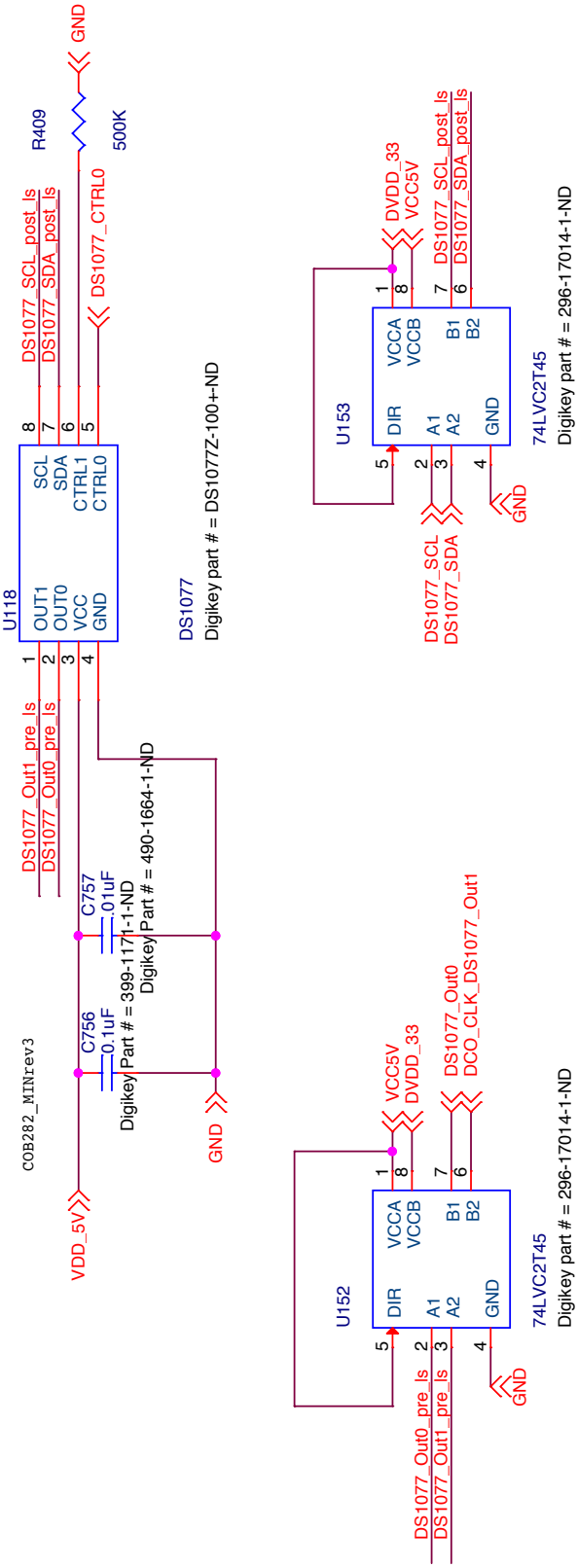


Figure 32: Clock portion of the RASP 3.0a PCB

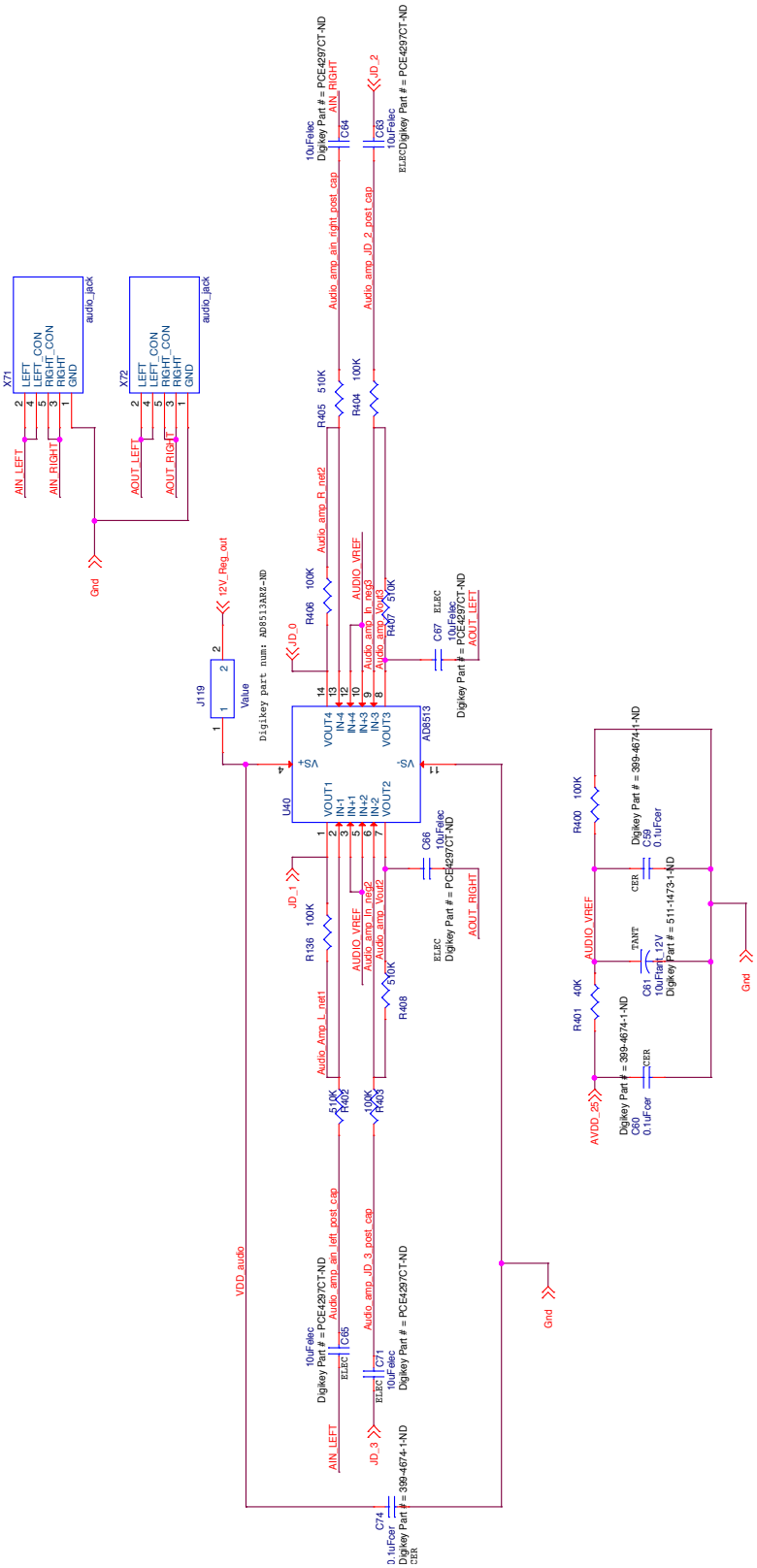


Figure 33: Audio portion of the RASP 3.0a PCB

Figure 34: Camera portion of the RASP 3.0a PCB

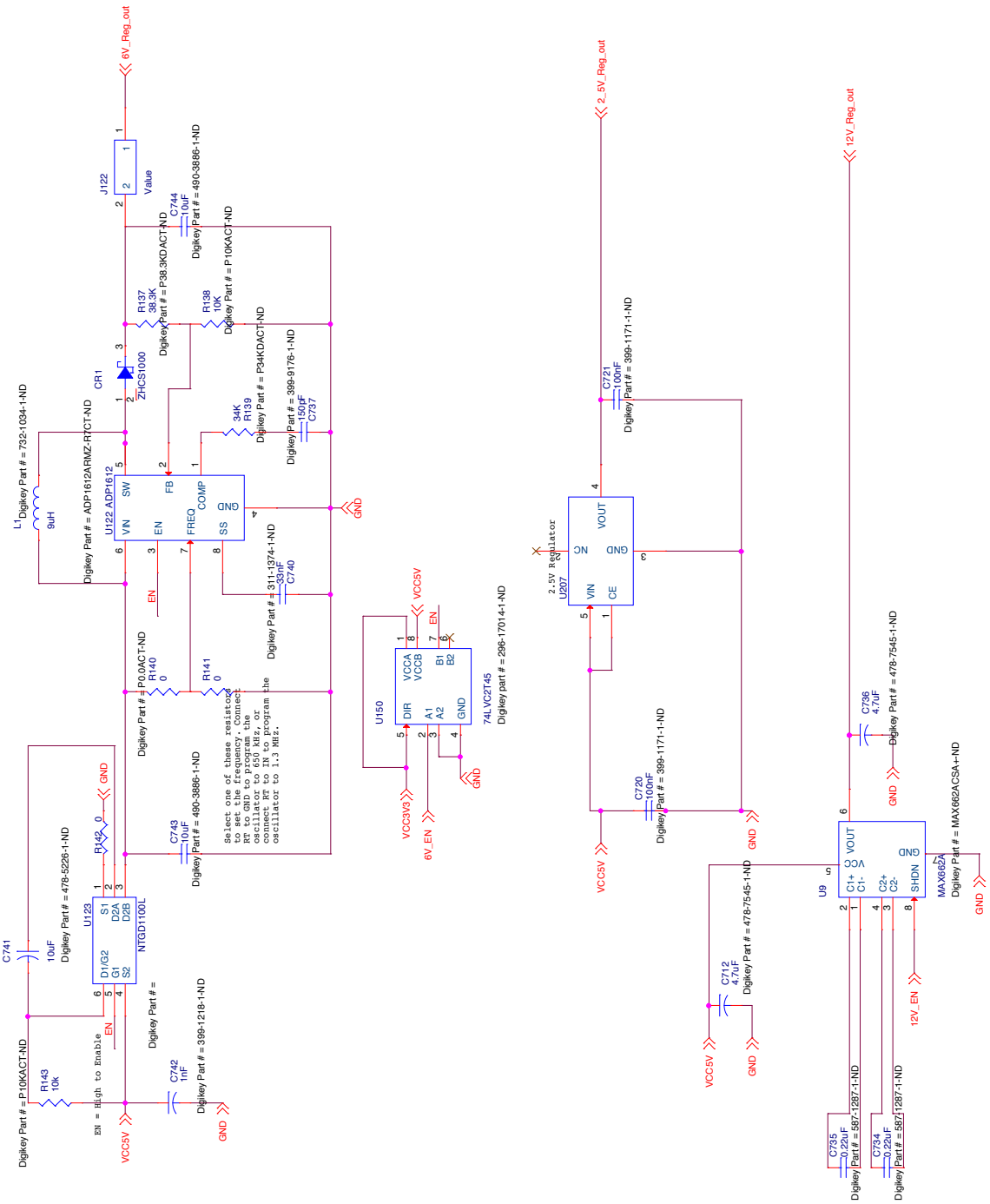


Figure 35: Power Regulation portion of the RASP 3.0a PCB

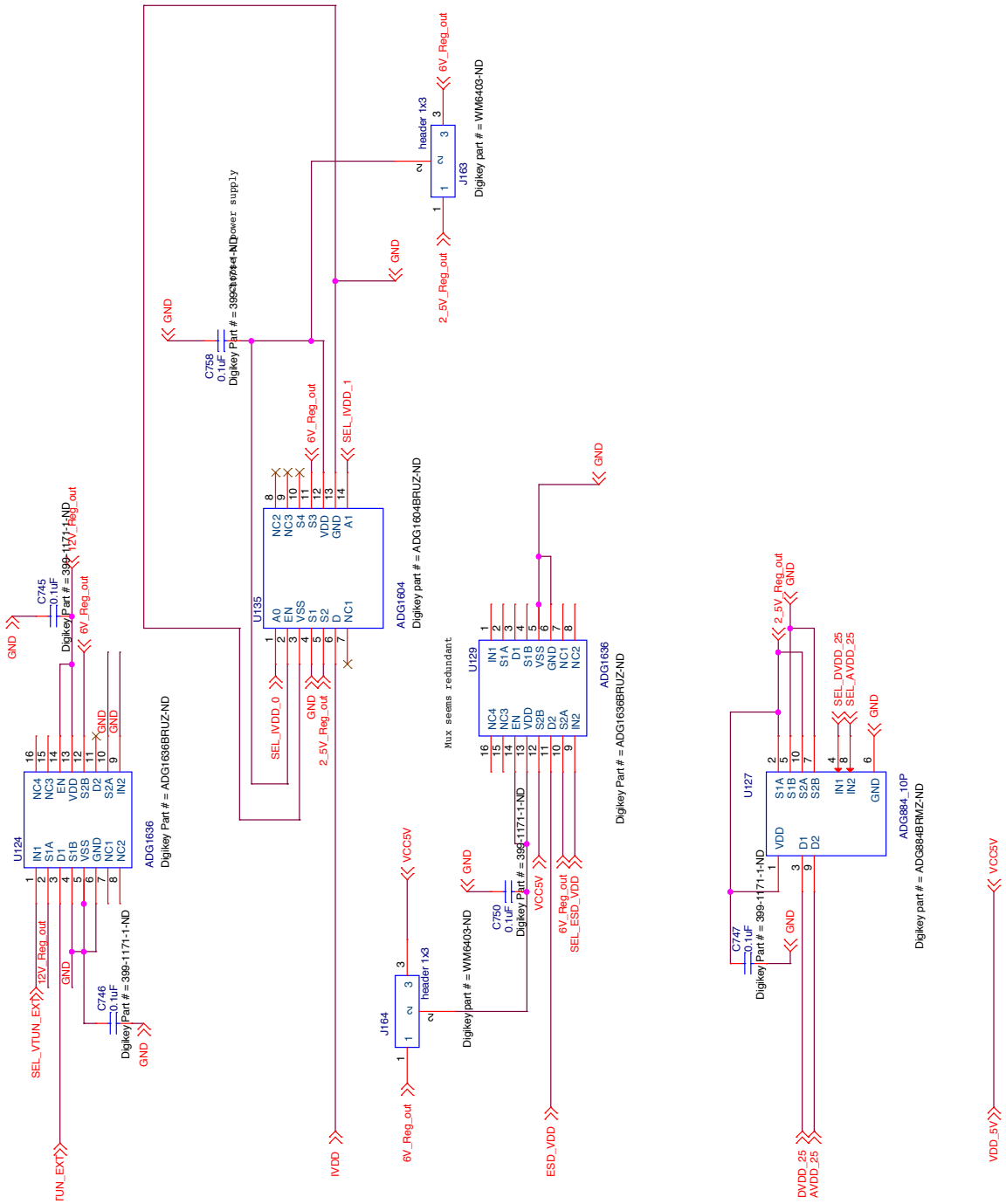


Figure 36: Power Switching portion of the RASP 3.0a PCB

APPENDIX E

RASP IC PCB PIN LAYOUT

The PCBs for the RASP 3.0a and RASP 3.0 chips have various header pins, surface-mount devices, and through-hole components as well as jumpers placed on them. Anyone besides the designer of the board would not know the precise location of signals such as ground (GND). The user would also not be aware of the placement of I/O pads to read output signals of circuits and to connect a device generating input voltages for a circuit. The users choose a numbered I/O pad in their design parameters, which corresponds to a I/O pad number on the pin layout diagram. Therefore, a meticulous layout of the boards for users was developed to reduce confusion of locating pin signals. Where distinguishing components are depicted to mimic the actual board's appearance, labels for all headers are inserted, and reference legends are provided for clarity.

The RASP 3.0a board that I led the design of is mostly utilized in the classroom, while the RASP 3.0 board is used internally within the research group at the moment due to the number of boards fully calibrated. The pin layout diagrams also depict the modifications made to the boards (*i.e.*, the addition of capacitors, resistors, and jumpers as well as the soldering of pins and the removal of pin connections) after fabrication during the calibration process as a form of visual documentation. The diagrams provide a sanity check that the latest improvements are reflected on the board being brought up to specifications. Similarly, if a user would like to change the jumper configuration or add jumpers, their decision is informed by the pin layout diagram.

The RASP 3.0 and RASP 3.0a have a different PCB layout. The RASP 3.0a chip

has specific I/O pad locations that are buffered, while others are not; however, the RASP 3.0 chip does not possess this attribute. The RASP 3.0 board is actually two PCBs connected together via a header and socket. Another difference is the RASP 3.0 board also utilizes a mini USB port. Whereas the RASP 3.0a is a single board that has a micro USB port, which has a lower profile and is more universal because a majority of electronic devices use it. An audio module consisting of an input and output audio jack from a previous board design was incorporated as well as a module for an imager sensor. A push button to reset the chip is also made available on the RASP 3.0a board.

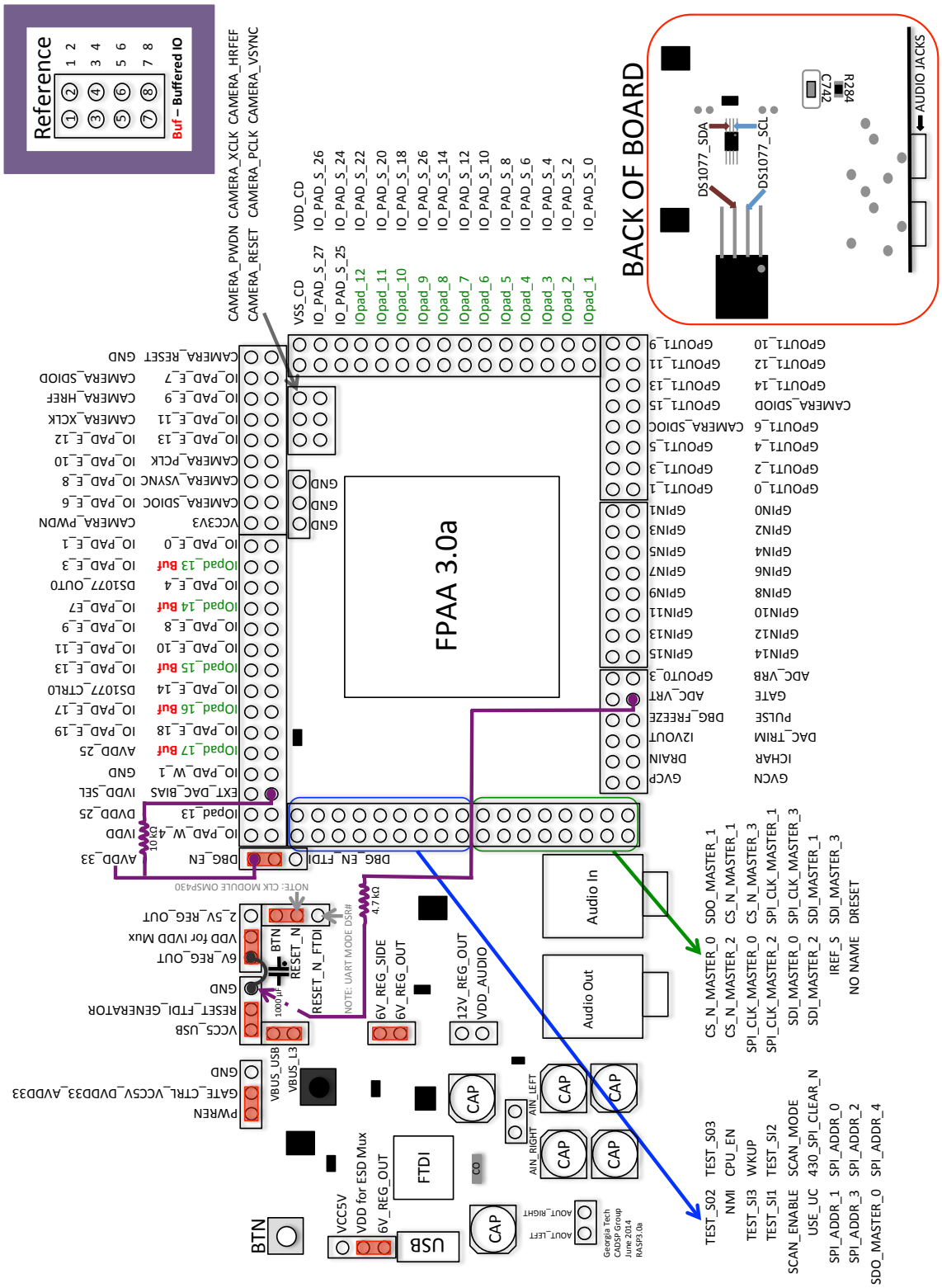


Figure 37: Class Board

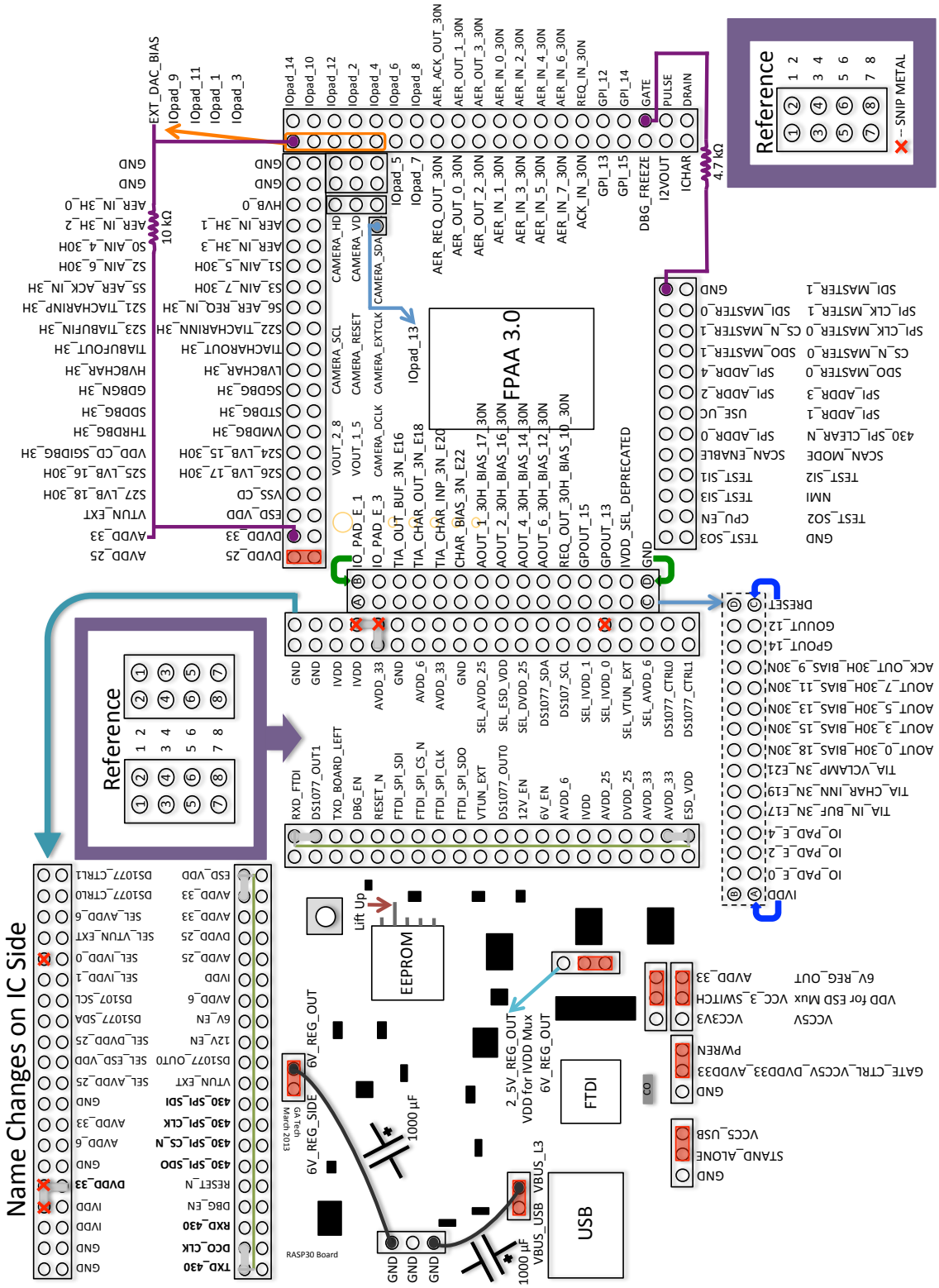


Figure 38: Research Board

REFERENCES

[1] Abdulwahed, M. and Nagy, Z. K., "Applying kolb's experiential learning cycle for laboratory education," *Journal of Engineering Education*, vol. 98, no. 3, pp. 283–294, 2009.

[2] Altera, "Simulation/Model - DSP Builder." `http://www.altera.com/products/software/products/dsp/dsp-builder.html`. [Online; accessed 10 September 2015].

[3] Altera, "SoC FPGAs: Integration to Reduce Power, Cost, and Board Size." `http://www.altera.com/devices/processor/soc-fpga/proc-soc-fpga.html`. [Online; accessed 10 September 2015].

[4] Astatke, Y., Chouikha, M. F., Connor, K. A., Ferri, A. A., Ferri, B. H., Meehan, K., Newman, D. L., Deyoe, M. M., and Walter, D. J., "Models of adoption and best practices for mobile hands-on learning in electrical engineering," in *Frontiers in Education Conference, 2013 IEEE*, pp. 511–513, IEEE, 2013.

[5] Auerbach, J. and Ferri, B., "Work in progress: The costs and benefits of using alternative approaches in lecture-based courses: Experience in electrical engineering," in *IEEE Frontiers in Education Conference (FIE)*, pp. T1E–1, IEEE, 2010.

[6] Basu, A., Brink, S., Schlottmann, C., Ramakrishnan, S., Petre, C., Koziol, S., Baskaya, F., Twigg, C. M., and Hasler, P., "A Floating-gate-based field-programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 1781–1794, September 2010.

[7] Basu, A., Brink, S., Schlottmann, C., Ramakrishnan, S., Petre, C., Koziol, S., Baskaya, F., Twigg, C. M., and Hasler, P., "A floating-gate-based field-programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 9, pp. 1781–1794, 2010.

[8] Becker, J. and Manoli, Y., "A continuous-time field programmable analog array (FPAA) consisting of digitally reconfigurable g m-cells," in *International Symposium on Circuits and Systems (ISCAS) Proceedings*, vol. 1, pp. I–1092, IEEE, 2004.

[9] Bishop, J. and Verleger, M., "Testing the flipped classroom with model-eliciting activities and video lectures in a mid-level undergraduate engineering course," in *IEEE Frontiers in Education Conference (FIE)*, pp. 161–163, IEEE, 2013.

[10] Bishop, J. L. and Verleger, M. A., "The flipped classroom: A survey of the research," *ASEE National Conference Proceedings, Atlanta, GA*, vol. 30, no. 9, 2013.

[11] Brink, S., Hasler, J., and Wunderlich, R., "Adaptive floating-gate circuit enabled large-scale fpaa," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 11, pp. 2307–2315, 2014.

[12] Cadence Design Systems, I., "EDA Tools and IP for System Design Enablement." `http://www.cadence.com/`. [Online; accessed 16 December 2015].

[13] Caverly, R. H., "A laboratory-oriented paradigm for undergraduate analog cmos microsystems design and education," in *Proceedings of the Tenth Biennial University/Government/Industry Microelectronics Symposium*, pp. 193–196, IEEE, 1993.

[14] Caverly, R. H., "Analog and mixed signal ic design via distance learning: The cad and laboratory dilemma," in *IEEE International Conference on Microelectronics Systems Education (MSE)*, p. 0083, IEEE, 2001.

[15] Caverly, R. H. and Zlatkovic, V., "Analog microsystem design education via the world wide web," in *IEEE International Conference on Microelectronics Systems Education (MSE)*, pp. 45–46, IEEE, 1999.

[16] Chang, S. T., Hayes-Gill, B. R., and Paull, C. J., "Multi-function block for a switched current field programmable analogue array," in *IEEE Midwest Symposium on Circuits and Systems*, vol. 1, pp. 158–161, IEEE, 1996.

[17] Collins, M., Hasler, J., and George, S., "Analog systems education: An integrated toolset and FPAA SoC boards," in *IEEE International Conference on Microelectronics Systems Education (MSE)*, pp. 32–35, IEEE, 2015.

[18] Collins, M., Hasler, J., and George, S., "An open-source tool set enabling analog-digital-software co-design," *Journal of Low-Power Electronics and Applications*, vol. 6, no. 1, p. 3, 2016.

[19] Collins, M., Hasler, J., and Shah, S., "An approach to using rasp tools in analog systems education," in *IEEE Frontiers in Education Conference (FIE)*, IEEE, 2016.

[20] Cooper, M. and Ferreira, J. M., "Remote laboratories extending access to science and engineering curricular," *IEEE Transactions On Learning Technologies*, vol. 2, no. 4, pp. 342–353, 2009.

[21] Cowan, G. E. R., Melville, R. C., and Tsividis, Y. P., *A VLSI analog computer/math co-processor for a digital computer*. University Microfilms, 2006.

[22] Digilent, "Analog Discovery 100MSPS USB Oscilloscope & Logic Analyzer." http://store.digilentinc.com/analog-discovery-100msps-usb-oscilloscope-logic-analyzer/. [Online; accessed 25 July 2016].

[23] Doboli, A. and Vemuri, R., "Exploration-based high-level synthesis of linear analog systems operating at low/medium frequencies," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 11, pp. 1556–1568, 2003.

[24] Donzellini, G. and Ponta, D., "A bottom-up approach to digital design with fpga," in *IEEE International Conference on Microelectronics Systems Education (MSE)*, pp. 31–34, IEEE, 2011.

[25] EZasic, "Architect and prototype analog & mixed-signal ICs." https://www.viadesigner.com/. [Online; accessed 16 December 2015; website is now https://ezasic.com/].

[26] Feisel, L. D. and Rosa, A. J., "The role of the laboratory in undergraduate engineering education," *Journal of Engineering Education*, vol. 94, no. 1, pp. 121–130, 2005.

[27] Ferri, B., Harris, J., Weitnauer, M. A., and Majerich, D., "A feedback-based approach for evolving a blended class model for large enrollment, multiple section circuits courses," in *Frontiers in Education Conference (FIE)*, pp. 1–8, IEEE, 2015.

[28] Ferri, B., Newstetter, W., and Majerich, D., "Instructor and graduate teaching assistant development and training for a blended linear circuits and electronics course," in *Frontiers in Education Conference (FIE)*, pp. 1–4, IEEE, 2015.

[29] Frantz, G., "Giving technology 2020 vision." Remarks by Gene Frantz at the TI Developer Conference, Washington, D.C.

[30] Frantz, G., "Digital signal processor trends," *IEEE micro*, vol. 20, no. 6, pp. 52–59, 2000.

[31] Ganesan, S. and Vemuri, R., "Analog-digital partitioning for field-programmable mixed signal systems," in *Conference on Advanced Research in VLSI (ARVLSI) Proceedings.*, pp. 172–185, IEEE, 2001.

[32] George, S., Kim, S., Shah, S., Hasler, J., Collins, M., Adil, F., Wunderlich, R., Nease, S., and Ramakrishnan, S., "A programmable and configurable mixed-mode FPAA SoC," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2253–2261, 2016.

[33] Graphics, M., "Analog/Mixed Signal IC Design." https://www.mentor.com/tannereda/ams-ic. [Online; accessed 16 December 2015].

[34] Hall, T., Twigg, C., Hasler, P., and Anderson, D., "Application performance of elements in a floating-gate FPAA," in *International Symposium on Circuits and Systems (ISCAS) Proceedings*, vol. 2, pp. II–589–92 Vol.2, May 2004.

[35] Hall, T. S., Twigg, C. M., Hasler, P., and Anderson, D. V., "Developing large-scale field-programmable analog arrays for rapid prototyping," *International Journal of Embedded Systems*, vol. 1, no. 3-4, pp. 179–192, 2005.

[36] Hasler, J., Kim, S., Shah, S., Adil, F., Collins, M., Koziol, S., and Nease, S., "Transforming mixed-signal circuits class through SoC FPAA IC, pcb, and toolset," in *European Workshop on Microelectronics Education (EWME)*, May 2016.

[37] Hasler, J., Shah, S., Kim, S., Lal, I., and Collins, M., "Remote FPAA system setup enabling wide accessibility of configurable devices," *Journal of Low-Power Electronics and Applications*, June 2016.

[38] Hasler, P., Scholttmann, C., and Koziol, S., "FPAA chips and tools as the center of an design-based analog systems education," in *IEEE International Conference on Microelectronics Systems Education (MSE)*, pp. 47–51, IEEE, 2011.

[39] Hasler, P., Scholttmann, C., and Koziol, S., "FPAA chips and tools as the center of an design-based analog systems education," in *IEEE International Conference on Microelectronics Systems Education (MSE)*, pp. 47–51, IEEE, 2011.

[40] Hirshfield, L. and Chachra, D., "Task choice, group dynamics and learning goals: Understanding student activities in teams," in *Frontiers in Education Conference (FIE)*, pp. 1–5, IEEE, 2015.

[41] Hong, L., Qian, K., Quan, G., and Ma, K., "Low-cost and portable labware for computing curriculum using scalable mobile sensory platform," in *IEEE Frontiers in Education Conference (FIE)*, pp. 200–202, IEEE, 2013.

[42] Huang, S. and Pierce, E., "The impact of a peer learning strategy on student academic performance in a fundamental engineering course," in *Frontiers in Education Conference (FIE)*, pp. 1–4, IEEE, 2015.

[43] Hudson, T. A., Copeland, B., and Solomon, D., "Creating a mixed-signal test and product engineering course," in *IEEE International Conference on Microelectronics Systems Education (MSE)*, pp. 56–59, IEEE, 2011.

[44] Kim, S., Hasler, J., and George, S., "Integrated floating-gate programming environment for system-level ICs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2015.

[45] Knight, C. D. and DeWeerth, S. P., "A shared remote testing environment for engineering education," in *IEEE Frontiers in Education Conference (FIE)*, pp. 1003–1006, IEEE, 1996.

[46] Lajevardi, P., Chandrakasan, A. P., and Lee, H.-S., "Zero-crossing detector based reconfigurable analog system," *IEEE Journal of Solid-State Circuits*, vol. 46, pp. 2478–2487, November 2011.

[47] Lanterman, A., Giardino, M., Ferri, B., Michaels, J., Hunt, W., and Ferri, A., "Embedding low-cost, portable experiments into a lecture-based signals and systems course," in *American Control Conference (ACC), 2014*, pp. 2543–2549, IEEE, 2014.

[48] Lazzaro, J., Ryckebusch, S., Mahowald, M. A., and Mead, C. A., "Winner-take-all networks of o (n) complexity," tech. rep., DTIC Document, 1988.

[49] Lee, E. K. and Gulak, P. G., "A cmos field-programmable analog array," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 12, pp. 1860–1867, 1991.

[50] Lee, E. K. and Gulak, P. G., "Field programmable analogue array based on mosfet transconductors," *Electronics Letters*, vol. 28, no. 1, pp. 28–29, 1992.

[51] Lin, J.-L., Imbertson, P., and Moore, T., "Classroom discourse development for" flipping classrooms": Theoretical concepts, practices, and joint efforts from engineering students and instructors," in *IEEE Frontiers in Education Conference (FIE) Proceedings*, pp. 1–8, IEEE, 2014.

[52] Linder, B., Somerville, M., Eris, Ö., and Tatar, N., "Work in progress–taking one for the team: Goal orientation and gender-correlated task division," in *IEEE Frontiers in Education Conference (FIE)*, pp. F4H–1, IEEE, 2010.

[53] Liu, H. J., *Archipelago-An Open Source FPGA with Toolflow Support*. PhD thesis, Master's Thesis, Electrical Engineering & Computer Sciences, University of California, Berkeley, CA, USA, 2013.

[54] Luu, J., Goeders, J., Wainberg, M., Somerville, A., Yu, T., Nasartschuk, K., Nasr, M., Wang, S., Liu, T., Ahmed, N., and others, "Vtr 7.0: Next generation architecture and cad system for fpgas," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 7, no. 2, p. 6, 2014.

[55] Ma, K.-S., Ma, Y., Wang, Y., Qian, K., Zheng, Q., and Hong, L., "Innovative mobile tool for engineering embedded design and security educations," in *Frontiers in Education Conference (FIE)*, pp. 1–4, IEEE, 2015.

[56] Maiti, A., Maxwell, A. D., Kist, A. A., and Orwin, L., "Merging remote laboratories and enquiry-based learning for stem education," *International Journal of Online Engineering*, vol. 10, no. 6, pp. 50–57, 2014.

[57] Mathworks, "FPGA Design and SoC Codesign." http://it.mathworks.com/solutions/fpga-design/. [Online; accessed 10 September 2015].

[58] May, D., Lensing, K., Tekkaya, A. E., Grosch, M., Berbuir, U., and Petermann, M., "What students use: Results of a survey on media use among engineering students," in *IEEE Frontiers in Education Conference (FIE) Proceedings*, pp. 1–6, IEEE, 2014.

[59] OpenCores, "OpenMSP430 Project: Open Core MSP430." http://opencores.org/projectopenmsp430. [Online; accessed 1 November 2015].

[60] Pankiewicz, B., Wojcikowski, M., Szczepanski, S., and Sun, Y., "A field programmable analog array for cmos continuous-time ota-c filter applications," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 2, pp. 125–136, 2002.

[61] Parent, D. W., Basham, E. J., Ng, S., and Weil, P. B., "An analog leaf cell for analog circuit design," in *IEEE International Conference on Microelectronics Systems Education (MSE)*, pp. 11–12, IEEE, 2005.

[62] Pierzchala, E., Perkowski, M. A., Van Halen, P., and Schaumann, R., "Current-mode amplifier/integrator for a field-programmable analog array," in *IEEE International Solid-State Circuits Conference (ISSCC), Digest of Technical Papers.*, pp. 196–197, IEEE, 1995.

[63] Queiroz-Neto, J. P., Sales, D. C., Pinheiro, H. S., and Neto, B. O., "Using modern pedagogical tools to improve learning in technological contents," in *Frontiers in Education Conference (FIE)*, pp. 1–8, IEEE, 2015.

[64] Ramakrishnan, S. and Hasler, J., "Vector-matrix multiply and winner-take-all as an analog classifier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 2, pp. 353–361, 2014.

[65] Rossi, D., Mucci, C., Pizzotti, M., Perugini, L., Canegallo, R., and Guerrieri, R., "Multicore signal processing platform with heterogeneous configurable hardware accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pp. 1990–2003, 2014.

[66] Rumberg, B. and Graham, D. W., "Reconfiguration costs in analog sensor interfaces for wireless sensing applications," in *International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 321–324, IEEE, 2013.

[67] Schlottmann, C. R. and Hasler, J., "High-level modeling of analog computational elements for signal processing applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pp. 1945–1953, 2014.

[68] Schlottmann, C. R., Shapero, S., Nease, S., and Hasler, P., "A digitally enhanced dynamically reconfigurable analog platform for low-power signal processing," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 9, pp. 2174–2184, 2012.

[69] Schlottmann, C. R., Shapero, S., Nease, S., and Hasler, P., "A digitally enhanced dynamically reconfigurable analog platform for low-power signal processing," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 9, pp. 2174–2184, 2012.

[70] Scilab-Enterprises, "scicos_model - Define a model structure." `https://help.scilab.org/docs/5.4.1/en_US/scicos_model.html`. [Online; accessed 4 February 2014].

[71] Scilab-Enterprises, "Scilab: Free and Open Source Software for Numerical Computation." `http://www.scilab.org/`. [Online; accessed 16 December 2015].

[72] Scilab-Enterprises, "sci_struct - Scicos block structure of a scilab computational function." `https://help.scilab.org/docs/5.4.1/en_US/sci_struct.html`. [Online; accessed 4 February 2014].

[73] Shah, S., Hasler, J., Kim, S., Lal, I., Kagle, M., and Collins, M., "Demonstration of a remote FPAA system for research and education," in *IEEE International Symposium onCircuits and Systems (ISCAS)*, pp. 319–324, IEEE, May 2016.

[74] Shryock, K. J., "Engaging students inside the classroom to increase learning," in *Frontiers in Education Conference (FIE)*, pp. 1–7, IEEE, 2015.

[75] Sivilotti, M. A., *Wiring Considerations in Analog VLSI Systems, With Application to Field-Programmable Networks (VLSI)*. California Institute of Technology, Pasadena, CA: Ph.D, 1991.

[76] Sousa, N., Alves, G. R., and Gericota, M. G., "An integrated reusable remote laboratory to complement electronics teaching," *IEEE Transactions on Learning Technologies*, vol. 3, no. 3, pp. 265–271, 2010.

[77] Svensson, L. and Peterson, L., "A system-level mixed-signal design course," in *IEEE International Conference on Microelectronics Systems Education (MSE)*, pp. 44–47, IEEE, 2015.

[78] Talbert, R., "Inverted classroom," *Colleagues*, vol. 9, no. 1, p. 7, 2012.

[79] Twigg, C. M. and Hasler, P. E., "Incorporating large-scale FPAAs in analog design courses," in *IEEE International Conference on Microelectronics Systems Education (MSE)*, pp. 171–172, June 2007.

[80] Twigg, C. M., Gray, J. D., and Hasler, P. E., "Programmable floating gate FPAA switches are not dead weight," in *IEEE International Symposium on Circuits and Systems*, pp. 169–172, IEEE, 2007.

[81] Twigg, C. M. and Hasler, P. E., "Incorporating large-scale FPAAs into analog design and test courses," *IEEE Transactions on Education*, vol. 51, no. 3, pp. 319–324, 2008.

[82] Weinhardt, M., Krieger, A., and Kinder, T., "A framework for pc applications with portable and scalable fpga accelerators," in *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, pp. 1–6, IEEE, 2013.

[83] Westerlund, T., Liljeberg, P., Plosila, J., and Tenhunen, H., "From traditional vlsi education to embedded electronics," in *IEEE International Conference on Microelectronics Systems Education (MSE)*, pp. 32–35, IEEE, 2013.

[84] Wolf, W. H., "Hardware-software co-design of embedded systems [and prolog]," *Proceedings of the IEEE*, vol. 82, no. 7, pp. 967–989, 1994.

[85] Wunderlich, R. B., Adil, F., and Hasler, P., "Floating gate-based field programmable mixed-signal array," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 8, pp. 1496–1505, 2013.

[86] Xilinx, "Zynq: All Programmable SoC Architecture." `http://www.xilinx.com/products/silicon-devices/soc.html`. [Online; accessed 10 September 2015].

[87] Zhao, Q., Amagasaki, M., Iida, M., Kuga, M., and Sueyoshi, T., "An automatic fpga design and implementation framework," in *International Conference on Field programmable Logic and Applications (FPL)*, pp. 1–4, IEEE, 2013.

[88] Zimmerman, B. J., Bandura, A., and Martinez-Pons, M., "Self-motivation for academic attainment: The role of self-efficacy beliefs and personal goal setting," *American Educational Research Journal*, vol. 29, no. 3, pp. 663–676, 1992.

VITA

Michelle D. Collins was born and raised in Willingboro, New Jersey, where she graduated from Willingboro High School in 2007 as class valedictorian. She earned her B.S. degree (Summa Cum Laude) in Electrical and Computer Engineering from the Morgan State University in 2011. As an undergraduate, she interned with the U.S. Army Corps of Engineers in Germany and the Johns Hopkins University Applied Physics Laboratory in Maryland. Michelle pursued her post-graduate studies at the Georgia Institute of Technology and she was awarded the university's President's Fellowship. She was a NSF Graduate Research Fellowship Program Fellow, an Alfred P. Sloan Foundation Minority Ph.D. Scholar, a Google Scholar, an Intel Fellow, and an ARCS Foundation Scholar. Michelle received her M.S. and Ph.D. degrees in Electrical and Computer Engineering in 2013 and 2016 from Georgia Tech, respectively.

Her research interests include CAD tools for configurable and programmable mixed-signal SoCs and engineering education. Michelle enjoys traveling, mentoring, coaching, volunteering, crocheting, and playing board games and card games in her leisure time.